Joachim Kock

Notes on Polynomial Functors

Very preliminary version: 2009-10-15 20:38
but with some mistakes fixed on 2016-08-16 15:09

PLEASE CHECK IF A NEWER VERSION IS AVAILABLE!
http://mat.uab.cat/~kock/cat/polynomial.html

PLEASE DO NOT WASTE PAPER PRINTING THIS!

Joachim Kock
Departament de Matemàtiques
Universitat Autònoma de Barcelona
08193 Bellaterra (Barcelona)
SPAIN

e-mail: kock@mat.uab.cat

VERSION 2009-10-15 20:38
Available from http://mat.uab.cat/~kock

This text was written in alpha. It was typeset in LaTeX in standard book style, with mathpazo and fancyheadings. The figures were coded with the texdraw package, written by Peter Kabal. The diagrams were set using the diagrams package of Paul Taylor, and with XY-pic (Kristoffer Rose and Ross Moore).

Preface

Warning. Despite the fancy book layout, these notes are in

VERY PRELIMINARY FORM

In fact it is just a big heap of annotations.

Many sections are very sketchy, and on the other hand many proofs and explanations are full of trivial and irrelevant details. There is a lot of redundancy and bad organisation. There are whole sections that have not been written yet, and things I want to explain that I don't understand yet. . . There may also be ERRORS here and there!

Feedback is most welcome.

There will be a real preface one day

I am especially indebted to André Joyal.

These notes started life as a transcript of long discussions between André Joyal and myself over the summer of 2004 in connection with [64]. With his usual generosity he guided me through the basic theory of polynomial functors in a way that made me feel I was discovering it all by myself. I was fascinated by the theory, and I felt very privileged for this opportunity, and by writing down everything I learned (including many details I filled in myself), I hoped to benefit others as well. Soon the subject became one of my main research lines and these notes began to grow out of proportions. The next phase of the manuscript was as a place to dump basic stuff I needed to work out in relation to my research. At present it is just a heap of annotations, and the reader is recommended to read rather the research papers that have seen the light in the interim:

- *Polynomial functors and polynomial monads* [39] with Gambino
- *Polynomial functors and trees* [63] (single-authored)

- *Polynomial functors and opetopes* [64] with Joyal, Batanin, and Mascari,
- *Local fibered right adjoints are polynomial* with Anders Kock,
- *Polynomials in categories with pullbacks* by Mark Weber,
- *Categorification of Hopf algebras of rooted trees* (by me),
- *Data types with symmetries and polynomial functors over groupoids* (by me),

(I will gradually incorporate the material from these papers into the notes in some form.)

Finally, a third aspect, which has been present all along, is the ambition that these notes can grow into something more unified — a book! I would like to survey connections with and applications to combinatorics, computer science, topology, and whatever else turns up. This will require of course that I learn more about these connections and applications, and I am gradually picking up these subjects. . .

I have benefited from conversations and email correspondence with Anders Kock, Andy Tonks, Clemens Berger, David Gepner, Ieke Moerdijk, Imma Gálvez, Jiri Adámek, Juan Climent Vidal, Krzysztof Kapulkin, Marek Zawadowski, Mark Weber, Martin Hyland, Michael Batanin, Neil Ghani, Nicola Gambino, Pierre Hyvernat, Tom Fiore, Tom Hirschowitz. . .

I am very thankful to Jesús Hernando Pérez Alcázar for organising my mini-course on polynomial functors in Santa Marta, Colombia, in August 2009 — a perfect opportunity for me to synthesise some of this material.

Barcelona, August 2009

JOACHIM KOCK
kock@mat.uab.cat

Contents

Preface iii

Introduction 1

Historical remarks 5

I Polynomial functors in one variable 9

0.0 Prologue: natural numbers and finite sets 11

1 Basic theory of polynomials in one variable 17

1.1 Definition 19

1.2 Examples 24

1.3 Other viewpoints 28

1.4 Basic operations on polynomials 30

1.5 Composition of polynomials (substitution) 34

1.6 Differential calculus 40

1.7 Properties of polynomial functors 43

2 Categories of polynomial functors in one variable 45

2.1 The category ***Set***$[X]$ of polynomial functors 46

Cartesian morphisms 48

2.2 Sums, products 54

2.3 Algebra of polynomial functors: categorification and Burnside semirings 56

2.4 Composition 60

2.5 The subcategory ***Poly***: only cartesian natural transformations 60

Products in ***Poly*** 63

Differentiation in ***Poly*** 65

3 Aside: Polynomial functors and negative sets **67**
3.1 Negative sets 67
3.2 The geometric series revisited 74
3.3 Moduli of punctured Riemann spheres 77

4 Algebras **81**
4.1 Initial algebras, least fixpoints 81
Functoriality of least fixpoints 84
4.2 Natural numbers, free monoids 84
4.3 Tree structures as least fixpoints 89
4.4 Induction, well-founded trees 93
4.5 Transfinite induction 95
4.6 Free-forgetful 99
Fixpoint construction via bar-cobar duality 104

5 Polynomial monads and operads **111**
5.1 Polynomial monads 111
Cartesian monads 111
The free monad on a polynomial endofunctor (one variable) 116
Examples 118
5.2 Classical definition of operads 120
5.3 The monoidal category of collections 121
5.4 Finitary polynomial functors and collections 124
Equivalence of monoidal categories 126
5.5 The free operad on a collection 127
5.6 *P*-operads 128

6 [Polynomial functors in computer science] **131**
6.1 Data types 131
Shapely types 139
6.2 Program semantics 141

7 [Species. . .] **145**
7.1 Introduction to species and analytical functors 145
7.2 Polynomial functors and species 146

II Polynomial functors in many variables 147

8 Polynomials in many variables 149
8.1 Introductory discussion 149
8.2 The pullback functor and its adjoints 152
Coherence 162
8.3 Beck-Chevalley and distributivity 163
8.4 Further Beck-Chevalley gymnastics 173
The twelve ways of a square 173
The six ways of a pair of squares 175
One more lemma 176
8.5 Composition 178
Rewrite systems and coherence 183
8.6 Basic properties 186
8.7 Examples 188
The free-category functor on graphs with fixed object set . . 190

9 Examples 193
9.1 Linear functors (matrices) 193
9.2 Finite polynomials: the Lawvere theory of comm. semirings 199
Lawvere theories 200
Proof of Tambara's theorem 205
9.3 Differential calculus of polynomial functors 208
Introduction 208
Partial derivatives 208
Homogeneous functors and Euler's Lemma 209
9.4 Classical combinatorics 214
9.5 Polynomial functors on collections and operads 217
The free-operad functor 217
Linear differential operators are linear 218
9.6 Bell polynomials 222

10 Categories and bicategories of polynomial functors 229
10.1 Natural transformations between polynomial functors . . . 229
Basic properties of ***PolyFun***(I, J): sums and products 238
Misc 238
Poly$^{c}(I, J)$: the cartesian fragment 239
Sums and products in ***Poly***$^{c}(I, J)$ 239

10.2 Horizontal composition and the bicategory of polynomial functors 240
Some preliminary exercises in the cartesian fragment 241
Horizontal composition of 2-cells 244

11 Double categories of polynomial functors 251
11.1 Summary 251
Reminder on double categories 253
The double category of polynomial functors 254
Lifts 260
old calculations 261
11.2 Horizontal composition 267
11.3 Cartesian 270
Horizontal composition of cartesian 2-cells 271
Misc issues in the cartesian fragment 273
Surjection-injection factorisation in ***Poly*** 273
Sums and products in the variable-type categories 274
Coherence problems 275

12 Trees (1) 277
12.1 Trees 280
12.2 From trees to polynomial endofunctors 281
Examples of trees 285
12.3 The category ***TEmb*** 288
12.4 P-trees 296

13 Polynomial monads 301
13.1 The free polynomial monad on a polynomial endofunctor . 301
13.2 Monads in the double category setting 306
relative 310
13.3 Coloured operads and generalised operads 310
13.4 P-spans, P-multicategories, and P-operads 310
Coloured operads 324
Polynomial monads and coloured operads 325

14 Trees (2) 327
14.1 P-trees and free monads 329
Examples of polynomial monads from trees 331
14.2 The category ***Tree*** 333

14.3 Trees of trees, constellations, and the Baez-Dolan construction 341

III Categorical polynomial functors 347
14.4 Introduction 349

15 [Polynomial functors for slices of *Cat*] 351
Cat is not locally cartesian closed 351
15.1 Conduché fibrations 353
15.2 Polynomial functors in *Cat* 361
15.3 The family functor 363
15.4 Final functors and discrete fibrations 366

16 [Polynomial functors for presheaf categories] 367
16.1 Some prelims 367
Kan extensions 367
Categories of elements 368
Nerves 373
Generic morphisms 375
Monads with arities 379
16.2 Distributors and mixed fibrations 385
16.3 The free-category monad 390
The free-multicategory monad 396
The free-coloured-operad monad 398
16.4 Local right adjoints 399

17 [Generalised species and polynomial functors] 403

18 Appendices 405
A Pullbacks 405

Index 419

Introduction

0.0.1 Polynomial functors. A polynomial with natural-number coefficients is a formal expression of the form

$$\sum_{n\in\mathbb{N}} a_n X^n \qquad\qquad a_n \in \mathbb{N},$$

like for example $4 + X^3 + 7\,X^8$. The notions involved—sums, products, and exponentiation—make sense also in the category of sets. We write $A + B$ for the disjoint union of two sets A and B, and we write $A \times B$ for their cartesian product. Finally we use the exponential notation X^E for the hom set $\mathrm{Hom}(E, X)$.[1]

Hence we can define a polynomial functor to be something like

$$\begin{array}{rcl} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set} \\ X & \longmapsto & A + B{\times}X^N + C{\times}X^R \end{array}$$

for some fixed sets A, B, C, N, R and a variable set X. A first idea would be to look at sums $\sum_{n\in\mathbb{N}} A_n \times X^n$, but it turns out to be very fruitful to forget about finiteness and look more generally at expressions like

$$\begin{array}{rcl} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set} \\ X & \longmapsto & \displaystyle\sum_{b\in B} X^{E_b}, \end{array} \tag{1}$$

where $(E_b \mid b \in B)$ is a family of sets, indexed by a set B.

[1] Since this notation is not very common outside category theory, let us explain why this is in fact a accurate picture of what is going on: if X is a set with 5 elements and E has 3 elements, then $\mathrm{Hom}(E, X)$ has 5^3 elements, as suggested by the notation X^E. Furthermore, if $E = E_1 + E_1$ (the disjoint union of sets E_1 and E_2), then $\mathrm{Hom}(E, X)$ is naturally isomorphic to $\mathrm{Hom}(E_1, X) \times \mathrm{Hom}(E_2, X)$. In other words, $X^{E_1+E_2} = X^{E_1} \times X^{E_2}$. (We will consistently use the equality sign for natural isomorphisms.)

Now it turns out that many of the things you can do with good old polynomials make sense also for polynomial functors: of course you can add them and multiply them, but you can also substitute them into each other or take derivative, and all these notions behave as expected. For example there is a Leibniz rule and a chain rule.

0.0.2 Some expected new features. Compared to polynomials of numbers, polynomials of sets have many new features. Many of them were to be expected just from the fact that sets are much righer than numbers: between sets there are maps, which allows for a much richer algebra than can exist for numbers. Notably between two polynomial functors there are natural transformations, an in particular a polynomial functor can have the structure of a monad, which will be a major theme. An important special case of a map between sets is the notion of symmetry: this leads to group theory, admittedly much richer than the theory of factorials!

0.0.3 Some unexpected new features. Next, it turns out that the categorical level also has a lot of features that one might perhaps not have guessed from the algebra of polynomial functions. One is the relationship with recursively defined sets and data structures. For example, the set of natural numbers can be characterised as the least solution to the polynomial equation of sets

$$X \cong 1 + X\,,$$

while the set of finite planar trees appears as least solution to the equation

$$X \cong 1 + \sum_{n \in \mathbb{N}} X^n.$$

This is about fixpoints for polynomial functors. This will be a major theme. Also applications to theoretical computer science.

Second, throughout we use a graphical language, which in fact is the crucial point in the close relationship with operads. That's another major theme, of course closely related.

Both these themes can be treated satisfactorily without too much categorical machinery, and we do so for pedagogical reasons.

0.0.4 Many-variables polynomials. However, to really get into these matters, it is necessary to study polynomial functors in many variables. We start that in Part II. At the same time we take the opportunity to upgrade to a more categorical setting.

Here is what polynomial functors in many variables are: given for example three variable sets X, Y, Z (i.e., a variable object in $\boldsymbol{Set}^3$), then a polynomial functor could be something like

$$\begin{array}{rcl} \boldsymbol{Set}^3 & \longrightarrow & \boldsymbol{Set}^2 \\ (X,Y,Z) & \longmapsto & (A \times X^M \times Y^N, X + B). \end{array}$$

That is, in analogy with polynomial maps $\mathbb{N}^n \to \mathbb{N}^m$ there are functors $\boldsymbol{Set}^I \to \boldsymbol{Set}^J$, for some fixed set of variables I and J. But here is what we will actually do:

A J-indexed family of polynomial functors in I-many variables has the form

$$(X_i \mid i \in I) \mapsto \Big(\sum_{b \in B_j} \prod_{e \in E_b} X_{s(e)} \mid j \in J \Big)\,, \tag{2}$$

equ:basicpol

where the indexing refers to the diagram of sets

$$\begin{tikzcd} I & E \arrow[l, "s"'] \arrow[r, "f"] & B \arrow[r, "t"] & J\,. \end{tikzcd} \tag{3}$$

equ:basicconf

This expression reduces to (1) when I and J are singleton sets. The functor specified in (2) is the composite of three functors: pullback along s, the right adjoint to pullback along f, and the left adjoint to pullback along t. The categorical properties of these basic kinds of functors allow us to manipulate polynomial functors like (2) in terms of their representing diagrams (3); this is a key feature of the present approach to polynomial functors.

(At this point we may as well take the step of abstraction of working in an arbitrary locally cartesian closed category $\mathscr{E}$. Even if the category $\mathscr{E}$ is not the category of sets (and in applications to logic and computer science $\boldsymbol{Set}$ is not typically a convenient model), the above formulae make sense if interpreted in the internal language of the locally cartesian closed category $\mathscr{E}$, which takes the form of a dependent type theory. (For this, one will perhaps need to assume that $\mathscr{E}$ is furthermore extensive — any topos will be all right.) This interplay, which goes back to the seminal paper of Seely [96], is an important aspect of the theory. A key feature of Martin-Löf type theory are the wellfounded trees (or W-types) which allow for recursive definitions and proofs by induction. Moerdijk and Palmgren [81] showed that the categorical counterpart of W-types are the initial algebras for polynomial functors in one variable, and Gambino and Hyland [38] extended this result to a correspondence between W-types in dependent type theory and initial algebras for polynomial functors in many variables; the passage to many variables also gives better closure properties. This interplay allows on one hand the formalism of polynomial functors to provide a clean semantics to fundamental constructions in type theory, and on the other hand, the internal language allows us to reason about abstract polynomial functors as if we were in the category of sets, and often the formulae look just like those for good-old polynomials of numbers.)

0.0.5 Trees. With the many-variable set-up we can define trees, and get a better hold on inductive data types and relations with operads. Base change is a key ingredient in the theory, and to handle it efficiently we assemble polynomial functors into a double category: the 2-cells in this double category are simple diagrams connecting diagrams like (3). In this setting we introduce a new formalism for trees: they are defined as polynomial functors satisfying a few axioms. It may seem a bit out of proportions that something relatively complicated as polynomial functors should serve to define something as fundamental as trees. It is better to view it like this: both trees and polynomial functors are defined in terms of diagrams of shape (3), and the fact that they are defined by this same shape explains why the relationship between them is so tight.

0.0.6 Categorical polynomial functors — incorporation of symmetries. In Part III, we start introducing symmetries into the polynomial functors. This could be done just by looking at discrete fibrations over groupoids, but we may as well develop the theory for general categories. The most useful form of this theory is that of presheaf categories. Now the left and right adjoints to the pullback functor are left and right Kan extension. This leads to a very useful notion of analytical functor (but we still prefer to call them polynomial, since they are still represented by diagrams like (3)). Many monads, whose algebras are various kinds of operads, are analytical.

Historical remarks

'Polynomial functor' should mean categorification of the notion of 'polynomial function'. Depending on which properties of polynomial functions one takes as guideline for the categorification, different notions result which might deserve to be called polynomial functors, and in the literature the name is in use for many different notions. On the other hand, since 'polynomial' is such a fundamental concept in mathematics, not surprisingly also categorified versions of it have been discovered and found use in many different contexts, and under different names.

The formalism of polynomial functors has proved useful in many areas of mathematics, ranging from algebra [77], [65] and topology [20], [90] to mathematical logic [43], [81] and theoretical computer science [52], [2], [47].

The theory of polynomial functors developed in these notes does not cover everything that has been called polynomial functor in the literature — notably our viewpoint does not immediately apply to linear contexts — but it is expressive enough to capture a wide range of situations in mathematics and computer science, and it provides a very convenient formalism. The natural scope of this formalism is that of locally cartesian closed categories, and in Part II we work in that settings. However, in Part I, for pedagogical reasons we stick to the category of sets. Overall, we do not include any finiteness condition in our definition of polynomial functor, but of course there are features specific to finitary or finite polynomial functors, and these will receive a special treatment.

Since the rough idea of 'polynomial functor' is so natural, it is not surprising that it has been discovered many times independently, and used to designate different notions. The first notions concerned the category of vector spaces or abelian groups. The earliest is perhaps Eilenberg–Mac Lane [33] back in the '50s. Their motivation is to calculate group homology. They studied $\boldsymbol{Ab}$-valued functors out of abelian categories preserving the zero object. They introduce the 'deviation' of such a functor, which is a sort of differential quotient, formally it is defined as a kernel of some sort. A functor is additive (also called linear) precisely when the deviation is zero. Otherwise, one can consider the deviation of the deviation, obtaining what are known as cross effects. A functor is called polynomial of degree n if the $(n+1)$ deviation is zero and the n deviation is non-zero.

Mention also Grothendieck (according to Joyal), Epstein–Dold. . .

Perhaps one can say polynomial functors were invented by Schur (without the name). Polynomial functors play an important role also in the representation theory of symmetric groups [77, Ch.1, Appendix]. Within that context, an endofunctor F of the category of finite dimensional vector spaces is said to be polynomial if for any V, W, the map $\mathrm{Hom}(V, W) \to \mathrm{Hom}(FV, FW)$ is a polynomial mapping (with respect to some and hence any basis). The usage of polynomial functors on vector spaces usually involve the actions of the symmetric groups and coincides with the linear species of Joyal [55]. However, there is also an important development of such symmetric polynomial functors on vector spaces over finite fields, in which case there are many new subtleties, due to the fact that different polynomials can define the same mappings in finite characteristics. There is a more general notion of polynomial functor (the previous ones are then called strict) which is related to the Steenrod algebra rather than with representations of the symmetric groups. We refer to [90] for an overview of the work in the area.

All these developments concern polynomial functors on vector spaces, and their motivation is representation theory and homological algebra.

Notions closer to the viewpoint of these notes surfaced gradually in the 1970s in the circle of ideas around algebraic theories, automata, and data types. The observation that signatures for algebraic structure (or for data types) define endofunctors and that the functorial properties of the latter contain important information about the structures was explored by the Prague school in the 1970s (see also Manes [78]), which is synthesised in Adámek and Trnkova's 1990 book *Automata and algebras in categories* [7]. Their works include conditions for existence of free algebras, existence of minimal realisation of machines, characterisation of languages recognisable by finite automata, and stressed fixpoint theory throughout. Polynomial functors form an important class of examples for these theories. Meanwhile the 1986 book of Manes and Arbib [79], *Algebraic approaches to program semantics* had become quite influential, and theoretical computer scientists began to explore categorical approaches to data types. Input came also from logic: Girard [43] studied polynomial functors under the name *normal functors* from the point of view of lambda calculus, and a categorical account of his theory, including simplifications and generalisations followed.

The objects in $\boldsymbol{Set}/\mathbb{N}$ are also called *signatures*. (This terminology comes from logic: each object is thought of as an abstract operation, and its image in $\mathbb{N}$ is its arity. The language is the set of all formulae that can be written with these operations.) Possibly this goes back to Philip Hall in the 1940s.

There is also the notion of *compositeur* of Lazard [73]. A *compositeur* is something very similar to an operad, and an *analyseur* is a compositeur with some sort of filtration. . .

Then there is the notion of *clone* (P. Hall), cf. Cohn's book *Universal Algebra* [28] (p.126 and 132). This is a little bit different than an operad: the list of operations to graft on top of the bottom operations are all required to be of the same arity n, and the resulting operation will be of this arity again: it is namely required to take n inputs, duplicating them and sticking them into the top operations in the appropriate slots(?).

The viewpoint central to these notes, that polynomial functors are represented by set maps, was discovered by Joyal in the 1980s, at the same time he discovered analytical functors. Joyal informs that he was unaware of Macdonald's book [77] when he wrote *Une théorie combinatoire des séries formelles* [54] in 1981, and became aware of it before *Foncteurs analytiques et espèces de structures* [55] (1985) (which cites [77]).

Analytical functors can be seen as polynomial functors with symmetries. The theory of species and analytical functors became a foundation for power series theory in combinatorics, see [54], [19]. Possibly, because of the success of analytic functors and species, the simpler theory of polynomial functors was never written down, and did not become mainstream.

Many results about polynomial functors are well known in other formulations. Quote Diers [32], Lamarche [67], Carboni–Johnstone [25]. That's the notion of familial representability. Large parts of Leinster's book [75] are in fact concerned with polynomial functors and notably polynomial monads, and implicitly establishes the relation with operad theory (of which Joyal was aware).

The importance in type theory was discovered by Moerdijk and Palmgren [81].

Also in operad theory and higher category theory [64], [63] (and implicitly in [11]).

Often the word 'polynomial' is also meant to imply finite degree, so as to ensure that it restricts to a functor with finite sets as inputs and values. A natural generalisation is to power series. The viewpoint taken here, which has been advogated by Joyal, is not to put any finiteness condition at all, acknowledging the category of abstract sets as the basic setting and allowing easy transfer to any topos, or indeed any locally cartesian closed category. The finiteness that some people may associate with the word 'polynomial' is just an artefact coming from the fact that only finite quantities can be expressed by numbers. The category of abstract sets was introduced by Cantor precisely as a framework for infinite quantities, and on the 'categorified' level the category of abstract sets is the natural basic setting for the polynomial idea.

So we take a rather liberal interpretation of what it means to be polynomial: for sums and products we do not limit ourselves to binary or finite operations, but rather let them be defined, respectively, in terms of left and right adjoints of the basic pullback functor. Hence we place ourselves in a locally cartesian closed category $\mathscr{E}$: this means that for every arrow $f : E \to B$ we dispose of three functors, the pullback functor $f^* : \textbf{\textit{Set}}/B \to \textbf{\textit{Set}}/E$, a left adjoint $f_!$, and a right adjoint f_*. By definition, a polynomial functor is one isomorphic to any composite of these three kinds of functors between slices of $\mathscr{E}$. By the Beck-Chevalley conditions for these adjoint pairs and by the fact that the right adjoint distributes over the left adjoint, every polynomial functor can then be brought on normal form, which means that it is represented by a diagram of form

$$I \leftarrow E \to B \to J$$

The functor is then the composite

$$\textbf{\textit{Set}}/I \overset{s^*}{\to} \textbf{\textit{Set}}/E \overset{p_*}{\to} \textbf{\textit{Set}}/B \overset{s_!}{\to} \textbf{\textit{Set}}/J$$

and polynomial functors can be characterised as those isomorphic to one of this form. The theory of polynomial functors, as presented here, is about manipulating polynomial functors in terms of their representing diagrams, in analogy to the characteristic feature of polynomial functions, that they can be manipulated in terms of their exponents and coefficients. At the risk of being presumptuous we like to compare this with the use of coordinates in linear algebra or in geometry. Since we are at one higher categorical level, it is not a question of reducing computations to manipulations with numbers, but rather about reduction to combinatorics. The importance of polynomial functors as a means to bring out the combinatorics was perhaps first noticed in [64], which used polynomial functors to establish the first purely combinatorial characterisation of the opetopes, the shapes underlying several approaches to higher category theory [75], starting with Baez-Dolan [11]. Throughout we shall use a graphical language of trees to reason with polynomial functors. This is not a whim: it will be justified in Chapters 12 and 14, where we'll see that there is a deep relationship between polynomial functors and trees.

Part I

Polynomial functors in one variable

0.0 Prologue: natural numbers and finite sets

THIS SEEMS TO BE AN INTRODUCTION TO SOMETHING ELSE.

Large parts of mathematics are concerned with reduction of geometry and topology to algebra, associating to geometrical objects various invariants that can be used to distinguish and compare the objects, up to some notion of equivalence depending on the context. However, to the man on the street, often only the reduced layer is visible, and math is conceived to be about algebraic manipulations with numbers (and occasionally an unknown X, a variable representing a number) — the conceptual background for the numbers is often lost.

It is instructive to revert this reduction process and try to look behind the numbers. Such an investigation can lead to conceptual explanations of 'facts of life', and reveal what is natural and what is not. This insight can guide our treatment of other mathematical problems.

$\boxed{\mathtt{N}}$ **0.0.1 Counting.** The simplest invariant is the cardinality of a finite set:

$$\begin{array}{rcl} \# : \textbf{\textit{FinSet}} & \longrightarrow & \mathbb{N} \\ X & \longmapsto & \#X \end{array}$$

Theorem. *Two finite sets are isomorphic if and only if they have the same cardinality.*

Really, this theorem is not much more than a tautology, because you could rightfully say that it contains only the defining property of the natural numbers. Baez and Dolan [12] tells the wonderful story about how shepherds invented the natural numbers as a device for comparing the size of two herds without having to line up the sheep in an explicit bijection (especially hard if the two herds you are comparing are the one of today with the one of yesterday, to see if everybody's there).

As algebraic geometers we may want to state the result in a slightly different manner, which is very useful:

Theorem-Definition. *The moduli problem of classifying finite sets up to isomorphism admits a fine moduli space, which we denote by* $\mathbb{N}$.

A *family* of finite sets is just a set map $p : E \to B$ with finite fibres (then the fibres $E_b := p^{-1}(b)$ are the members of the family). The base set B is not required to be finite.

To say $\mathbb{N}$ is a fine moduli space means that it carries a universal family $u : \mathbb{N}' \to \mathbb{N}$: for any family of finite sets $p : E \to B$, there is a unique map $\kappa : B \to \mathbb{N}$ (called the *classifying map*), such that the family p is a pullback along κ of the universal family:

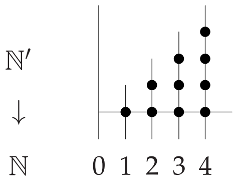
$$\begin{tikzcd}
E \arrow[r, "\kappa'"] \arrow[d, "p"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N}' \arrow[d, "u"] \\
B \arrow[r, "\kappa"'] & \mathbb{N}
\end{tikzcd}$$

(Note that κ' is not unique, cf. 2.1.9!) Now the invariant # is described like this: to a finite set S associate the image under the classifying map of the trivial family $S \to 1$.

Let us exhibit a universal family:[2] it is just a map of sets $u : \mathbb{N}' \to \mathbb{N}$ whose fibre over n is a finite set with n elements. We set $\mathbf{n} := \{0, 1, \ldots, n-1\}$. One concrete description of the universal family is this:

$$\mathbb{N}' = \sum_{n \in \mathbb{N}} \mathbf{n} = \{(n, i) \in \mathbb{N} \times \mathbb{N} \mid i < n\}.$$

Then $\mathbb{N}' \to \mathbb{N}$ is simply the projection $(n, i) \mapsto n$:



0.0.2 Commutative semirings and distributive categories. In $\mathbb{N}$ we have the arithmetic operations of addition and multiplication, and these satisfy a list of familiar rules, which amount to saying that $(\mathbb{N}, +, 0, \times, 1)$ is a commutative *semiring* — that's like a commutative ring, but without requiring the existence of additive inverses.

Looking back, these operations are in fact *defined* in terms of operations existing on the level of finite sets: for example, the natural number $a + b$ is defined as the cardinality of the disjoint union of A and B, where A is a

[2] Perhaps this should not be called a universal family, in view of the non-uniqueness of κ'. It is a universal family in the sense that it represents the functor which to a given set B associates the set of isomorphism classes of families of finite sets of B.

set of cardinality a and B is a set of cardinality b. Similarly, multiplication is defined in terms of the cartesian product of finite sets. This makes sense because disjoint union and cartesian product are functorial constructions, so if you replace A or B by isomorphic sets you get an isomorphic result. Furthermore, disjoint union and cartesian product of sets are characterised by universal properties (see for example [12]), which in turn imply that associativity as well as commutativity hold (up to canonical isomorphism), and hence explain these laws for the natural numbers. Note also that (up to canonical isomorphism) the empty set 0 (the initial set) is neutral for disjoint union, and the singleton set 1 (the terminal set) is neutral for the cartesian product.

The main theme of this exposition is to lift arithmetics from natural numbers to finite sets, and in fact we will use arbitrary sets. To make this lifted arithmetics look as familiar as possible, we write $A + B$ for the disjoint union of sets A and B, and $\sum_{b \in B} E_b$ for the disjoint union of a family of sets $(E_b \mid b \in B)$ indexed by a (possibly infinite) set B. (And as usual we write $\times$ for the binary cartesian product and $\prod$ for an arbitrary, possibly infinite, product.)

0.0.3 Distributive category and Burnside semiring. The construction of the semiring $\mathbb{N}$ from ***FinSet*** has a natural generalisation which is useful: by definition, a *distributive category* is a category $\mathscr{D}$ where finite sums and finite products exist (in particular there is an initial object and a terminal object) and the distributive law holds, i.e. the natural morphism

$$A \times X + A \times Y \longrightarrow A \times (X + Y) \tag{4}$$

$\boxed{\texttt{distr}}$

is an isomorphism.[3] The set of isomorphism classes of $\mathscr{D}$ inherits from $\mathscr{D}$ the structure of a commutative semiring, called the *Burnside semiring* of $\mathscr{D}$. So we can summarise our findings so far by saying that

$\mathbb{N}$ *is the Burnside semiring of* ***FinSet****.*

In 2.3.4 we shall see that the polynomial semiring $\mathbb{N}[X]$ is the Burnside semiring of the category ***FinSet***$[X]$ of finite polynomial functors, and in Section 3 we shall have a short look at the mythical *negative sets* whose Burnside semiring would be $\mathbb{Z}$.

[3] We ought to require also that $A \times \emptyset \simeq \emptyset$, but I seem to recall that Cockett has shown that this condition is automatic from the first one! – perhaps in [26].

0.0.4 Exponentiation and distributivity. The categories ***FinSet*** and ***Set*** sport an extra operation which is crucial: exponentiation. Given two sets X and E, we set

$$X^E := \mathrm{Hom}_{\boldsymbol{Set}}(E, X).$$

Note that if $\#X = x$ and $\#E = e$, then $\#(X^E) = x^e$, which justifies the notation. (In fact, it would be more precise to say that exponentiation in $\mathbb{N}$ is *defined* by this equation.) Note also the natural isomorphisms

$$X^{E_1+E_2} = X^{E_1} \times X^{E_2}, \qquad (X_1 \times X_2)^E = X_1^E \times X_2^E,$$

which are immediate consequences of the universal properties of disjoint union and cartesian product, respectively. Here and throughout we use the equality sign for natural isomorphisms. This effectively explain these laws for exponentiation in $\mathbb{N}$.

Exponentiation is right adjoint to multiplication:

$$\frac{A \times E \longrightarrow X}{A \longrightarrow X^E}$$

This fact implies distributivity: the inverse to (4) comes about like this:

$$\begin{array}{c}
A \times (X+Y) \longrightarrow A{\times}X \;+\; A{\times}Y \\
\hline
X+Y \longrightarrow (A{\times}X \;+\; A{\times}Y)^A \\
\hline
X \;\to\; (A{\times}X \;+\; A{\times}Y)^A, \qquad Y \;\to\; (A{\times}X \;+\; A{\times}Y)^A \\
\hline
\underbrace{A \times X \;\to\; A{\times}X \;+\; A{\times}Y, \qquad A \times Y \;\to\; A{\times}X \;+\; A{\times}Y}_{\text{the sum injections}}
\end{array}$$

and hence explains the distributive law in $\mathbb{N}$.

0.0.5 Equations and isomorphisms — bijective proofs. So we can ask which equations valid in $\mathbb{N}$ come from finite sets? Which can be lifted or *categorified*? This question has a long history in combinatorics: combinatorists speak of a bijective proof of a combinatorial identity, if each side of the equation counts something and the equation is proved by establishing a explicit bijection between the two sets being counted. There are identities which are known to be true, but for which no bijective proof is known. . .

0.0.6 Categorification. The example is part of a programme known under the name *categorification* which is this process: substitute sets by categories in such a way that equations between elements in the set becomes isomorphisms between objects. As we shall see it is sometimes necessary more generally to replace equations by *equivalences*, not just isomorphisms. In any case the goal of this process is to understand not only whether an equation holds or not, but also *why* it holds. The word 'categorification' was introduced by Crane [30] and the programme has been advocated much by Baez and Dolan (see for example [12] for a very readable account, very relevant to the themes of the present exposition). Previously Lawvere and Schanuel had used the term 'objectification'.

There are many other reasons why this process is fruitful. Generally speaking it is because sets are much richer than numbers, and can interact in more interesting ways. Specifically, there are mappings between sets! As a particular case, sets have symmetries. For example, a finite set A of n elements has a symmetry group, $\mathrm{Aut}(A) \simeq \mathfrak{S}_n$, which is a very interesting algebraic object associated to A, much more interesting than associating a number $n!$ to a given number n. So from this viewpoint, the categorified setting gives us all of group theory, whereas in contrast, the reduced setting just gives us the theory of factorials (and their prime factors).

More generally, a set can have structure. In addition to group structure, it could be an ordering, a vector space structure, a topology and so on. Furthermore, things you can do with usual constant sets, you can often do with variable sets too [72], or more generally in a topos. For example, the theory of polynomial functors is closely related to the Sierpiński topos, cf. Section 2.

Finally, to come back the the concrete question of natural numbers and finite sets, we notice that many things we can do with finite sets, we might as well do with infinite sets! This was Cantor's original motivation for inventing abstract sets, to handle infinite quantities. E.g. for numbers it is often a problem that an infinite sum $\sum_{i=0}^{\infty} a_i$ is not necessarily a well-defined number. But an infinite sum of sets, well that's just another set. There is some hope that categorification can help physics overcome problems related to ill-defined sums that tend to appear, for example in quantum mechanics, which are currently handled by dubious renormalisation tricks.

Recently, motivation has come also from higher category theory. . .

Chapter 1

Basic theory of polynomials in one variable

1.0.1 Reminders on polynomials with natural-number coefficients. A polynomial with natural-number coefficients is a formal expression of the form

$$\sum_{n\in\mathbb{N}} a_n X^n \qquad\qquad a_n \in \mathbb{N},$$

like for example $4 + X^3 + 7\,X^8$. The symbol X is a formal variable, a mere place holder. So formally one could also say that a polynomial is a finite list of natural numbers $(a_0, a_1, a_2, \ldots, a_k)$, but the idea is of course that one can substitute numbers into X, and hence a polynomial defines a function $\mathbb{N} \to \mathbb{N}$. Two polynomials define the same function if and only if their sequences of coefficients agree. This functional interpretation immediately explains the familiar operations of addition and multiplication of polynomials, with which the set of all polynomials becomes a commutative semiring,[1] denoted by $\mathbb{N}[X]$.

From the function viewpoint there is also an obvious operation of substituting one polynomial into another—this is nothing but composition of functions—and by repeated use of the distributive law (which holds in $\mathbb{N}$ and therefore in the function semiring $\mathbb{N}[X]$ too), it is easy to see that the composite of two polynomial functions is again given by a polynomial.

[1] Also called a *rig* by some authors — this is not recommended since it is not easy to translate into other languages.

There is also the operation of differentiation: it is defined in the usual way by

$$\begin{array}{rcll} D : \mathbb{N}[X] & \longrightarrow & \mathbb{N}[X] & \\ X^n & \longmapsto & nX^{n-1}, & n > 0 \\ X^0 & \longmapsto & 0 & \end{array}$$

and extending $\mathbb{N}$-linearly. (When polynomials are interpreted as functions $\mathbb{N} \to \mathbb{N}$, differentiation has no direct analytical interpretation, but of course the polynomials could also be interpreted as functions of a real variable.) We have the usual Leibniz rule and chain rule:

$$\begin{array}{rcl} D(F \cdot G) & = & DF \cdot G + F \cdot DG \\ D(F \circ G) & = & (DF \circ G) \cdot DG. \end{array}$$

1.0.2 Polynomial functors — overview. The notions involved—sums, products, and exponentiation—make sense also in the category of sets. Hence a polynomial functor will be something like

$$\begin{array}{rcl} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set} \\ X & \longmapsto & A + B \times X^N + C \times X^R \end{array}$$

for some fixed sets A, B, C, N, R and a variable set X. Many of the things you can do with good old polynomials make sense also for polynomial functors: of course you can add them and multiply them, but you can also substitute them into each other or take derivative, and all these notions behave as expected. For example there is a Leibniz rule and a chain rule. In a precise sense, polynomial functors are the categorification of polynomials with natural-number coefficients.

There is also a notion of polynomial functor in many variables, but it is better treated with some more categorical machinery, so it is postponed to Part II of these notes.

In this chapter we study those elementary operations on polynomial functors. But that is only a beginning: soon we'll see that the ***Set*** version of polynomials is much richer, and has connections to many areas of mathematics we might not have guessed just from the elementary algebra of polynomials of numbers.

1.1 Definition

1.1.1 Notation and conventions. We use the symbol $+$ and $\sum$ for the disjoint union of sets. We denote the empty set by 0 (although occasionally the usual notation $\emptyset$ is also practical, as well as $\mathbf{0}$ for the empty ordinal).

If $p : E \to B$ is a map of sets, we denote by E_b the fibre $\{e \in E \mid p(e) = b\}$, and we make the natural identification

$$\sum_{b \in B} E_b = E.$$

(Formally, the disjoint union $\sum_{b \in B} E_b$ is defined to be the set of pairs (b, e) with $b \in B$ and e such that $p(e) = b$. But since all the sets E_b are subsets in the same set E, and since they are clearly pairwise disjoint, is is natural to replace the 'disjoint union' by the 'union inside E', hence arriving at the identification.)

Another identification we make freely is:

$$\sum_{b \in B} X = B \times X$$

In this case the definition of disjoint union quickly gives the result. (The proof involves $(B \times X)_b = \{b\} \times X = X$.)

Similarly,

$$\prod_{e \in E} X = X^E.$$

Here X^E denote the set of all set maps from E to X.

1.1.2 Monomials. The simplest polynomials are the (monic) monomials. In one variable, these are functions of the form

$$\begin{array}{ccc} \mathbb{N} & \longrightarrow & \mathbb{N} \\ x & \longmapsto & x^n. \end{array}$$

Correspondingly, we can consider sets of the form X^E:

By definition, a *monomial functor* is a functor

$$\begin{array}{ccc} \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & X^E \end{array}$$

for some fixed set E. In other words, it is just a representable endofunctor on ***Set***, represented by E. If $\varphi : X \to Y$ is a map of sets, it's image is the map $\varphi^E : X^E \to Y^E$ taking $E \to X$ to $E \to X \to Y$.

While a monomial is specified by a single set, of course a sum of monomials will be specified by a family of sets.

1.1.3 Polynomials. A *polynomial functor* (in one variable) is by definition a functor ***Set*** $\to$ ***Set*** (isomorphic to one) of the form

$$X \mapsto \sum_{b \in B} X^{E_b}$$

where E_b is a family of sets indexed by some other set B. The value of the functor on a map of sets $\varphi : X \to Y$ is the map

$$\sum_{b \in B} \varphi^E : \sum_{b \in B} X^E \to \sum_{b \in B} Y^E$$

defined term-wise by taking $E_b \to X$ to $E_b \to X \to Y$.

To give a family of sets $(E_b \mid b \in B)$ amounts to giving a set map

$$p : E \to B,$$

and then E_b denotes the fibre $p^{-1}(b)$. This is just to write $E = \sum_{b \in B} E_b$. We say that the family $p : E \to B$ *represents* the polynomial functor.

We do not require the indexing set B to be finite (so in this sense we are allowing polynomials with infinitely many terms), but sometimes it will be convenient to require all fibres to be finite. Such a map will be called *finitary*. This corresponds to allowing only finite exponents.

So to every set map $p : E \to B$ there is associated a polynomial functor

$$\begin{array}{rcl} P : \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & P(X) = \sum_{b \in B} X^{E_b}. \end{array}$$

To give a single set is then just to give a map $E \to \mathbf{1}$, showing that a monomial is just special case of a polynomial.

Conversely, given a polynomial functor (something of the form $X \mapsto \sum_{b \in B} X^{E_b}$ then there is a unique map of sets $E := \sum_{b \in B} E_b \to B$. SHOULD

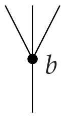
ANALYSE A LITTLE BIT HOW CANONICAL THIS IS. FOR EXAMPLE, FOR LINEAR FUNCTORS (CF. BELOW) THERE SEEMS TO BE NOT VERY MUCH HOLD ON THE SET E...

Convention: we use the word *polynomial functor* for the functors, while we shall sometimes say *polynomial* for the configuration of sets specified by $E \to B$. This is perhaps not so useful here in the one-variable case...

graphical1

1.1.4 Graphical interpretation. The whole point about the theory of polynomial functors is to manipulate $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$ in terms of the representing map $p : E \to B$. It turns out this is intimately linked with the combinatorics of trees. The key to understanding these relations is the graphical interpretation we now present. We shall use trees as a very convenient way to manipulate polynomial functors. Later, in Chapter 14, we shall see that this connection is in fact fundamental: we shall see that trees *are* polynomial functors (in several variables) and that polynomial functors can be described as colimits of trees. (That discussion is postponed only because it needs many-variables theory.)

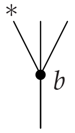
The important property of an element in B is really the corresponding fibre E_b, so we will build it into the picture: we picture an element $b \in B$ by drawing b itself as a dot, and represent the fibre E_b as a set of edges coming into the dot:



We have also drawn an edge coming out of the dot. There is no justification at this point of the exposition for having this edge in the picture, but we are going to need it crucially later on: when we come to polynomials in many variables, we will need this root edge to carry a label specifying which variable we refer to. We think of b as an operation. We think of the edges above the node as input slots and we think of the root as the output. This is another reason why we insist on drawing the root edge —we want every operation to have an output.

The input slots will eventually be filled: this happens when we evaluate the polynomial functor on a set, as we shall see shortly.

To specify an element in E is the same thing as specifying a fibre and then an element in that fibre, i.e., specifying $b \in B$ together with an element in E_b. This is just a paraphrasing of the canonical bijection $E = \sum_{b \in B} E_b$. So we can represent an element in E as



where the mark indicates which element in the fibre we chose.

constant

1.1.5 Constant functors. Consider the family given by the map $0 \to B$. The corresponding polynomial functor is

$$X \mapsto \sum_{b \in B} E^0 = \sum_{b \in B} 1 = B,$$

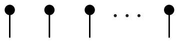
the constant functor $X \mapsto B$. As a special case, the map $0 \to 0$ represents the constant zero polynomial $X \mapsto 0$. Another important case is $0 \to 1$, representing the constant polynomial $X \mapsto 1$.

This gives us a way of interpreting any set as a constant polynomial functor, just like we can interpret a natural number as a constant polynomial. (We shall see later that this defines a functor from ***Set*** to the category of polynomial functors; it is in fact a homomorphism of semiring categories (2.3.5).)

We like to use the symbol X as a place holder, a position for plugging in any given set. Let us now consider a fixed set S. We picture it as a bunch of dots, one for each element:

$$S = \{\bullet \quad \bullet \quad \cdots \quad \bullet\}$$

We can also interpret S as the constant polynomial functor $X \mapsto S$, it is represented by the map $\emptyset \to S$. Hence all the fibres are empty, so all operations of this polynomial functor are nullary. So in this case we picture the elements as

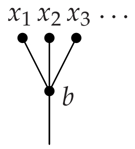


So when we are picturing the elements of S as dots, we are also thinking of them as nullary operations. This is standard: we think of static sets as sets of nullary operations for the corresponding constant functor.

P(X)

1.1.6 Evaluating polynomial functors. Given a polynomial functor $P(X) = \sum_{b \in B} X^{E_b}$, the image set comes with a natural projection to B, since it is a sum indexed by the elements of B. (This is to say that actually P factors as $\boldsymbol{Set} \to \boldsymbol{Set}/B \to \boldsymbol{Set}$ where the last functor is the forgetful functor that forgets about the map to B.) The fibre over $b \in B$ is the set of maps $E_b \to X$. Since we have already chosen b at this point of the description we have the bouquet b, and giving the map $E_b \to X$ amounts to decorating each leaf with an element of X.

In graphical terms, for a fixed set S, the value set $P(S)$ can be pictured as the set of bouquets like for B, but where each leaf is decorated by an element in S:



Note that repetition may occur in such a decoration, so in the picture the x_i are not assumed to be distinct.

In general we distinguish between *labelling* (which means that the labels are unique) and *decorations*, where repetition can occur. In other words, labellings are bijections, decorations are maps.

We have already mentioned that S can be identified with the constant polynomial functor with value S. We will soon define the composition of two polynomial functors, and as such the set $P(S)$ will be identified with the composite polynomial functor $P \circ S$, which is constant since the first factor is. The graphical interpretations of these two viewpoints match.

If $\varphi : X \to Y$ is a map of sets, it is also easy to grasp the image map $P(\varphi)$: it is the map from X-decorated bouquets to Y-decorated bouquets, replacing each decoration x by $\varphi(x)$.

1.1.7 Sneak preview of many-variable polynomial functors. They are given by diagrams $I \leftarrow E \to B \to J$. Polynomial functors in one variable are given by diagrams $1 \leftarrow E \to B \to 1$, and hence are really functors $\boldsymbol{Set}/1 \to \boldsymbol{Set}/1$. Now a constant set S we are saying can be interpreted as a functor $1 \to \boldsymbol{Set}$. This can also be seen as the polynomial functor in no variables given by

$$0 \leftarrow 0 \to S \to 1$$

observing that ***Set***$/0 = 1$. In contrast, a constant polynomial functor ('in one variable') is given by

$$1 \leftarrow 0 \rightarrow S \rightarrow 1.$$

P(1)

1.1.8 Examples. In particular there is a natural bijection

$$P(\mathbf{1}) = \sum_{b \in B} 1^{E_b} = \sum_{b \in B} 1 = B.$$

You can interpret this by saying that the bottom set B is the sum of all the coefficients.

We will often consider the special set $1 = \{\texttt{blank}\}$. If we feed this set into P the result is the set of bouquets with each leaf decorated by `blank`. The idea is of course that we leave the leaf undecorated, so in conclusion $P(1) = B$.

1.2 Examples

MAKE DRAWINGS OF EACH OF THE EXAMPLES!

1.2.1 Notation. For each $n \in \mathbb{N}$, denote by $\mathbf{n}$ the set $\{0, 1, 2, \ldots, n-1\}$. In particular $\mathbf{0}$ is the empty set, and $\mathbf{1}$ is the singleton set (terminal set).

1.2.2 Constant functors. We saw that the map $0 \rightarrow B$ represents the constant polynomial functor $X \mapsto B$. More generally, given any polynomial functor $P(X) = \sum_{b \in B} X^{E_b}$, we can evaluate at the empty set and find

$$P(0) = \sum_{b \in B} 0^{E_b},$$

but the set of maps $E_b \rightarrow 0$ is empty unless E_b is empty, in which case there is exactly one map. So if we set $B_0 = \{b \in B \mid E_b = 0\}$, we find

$$P(0) = B_0,$$

the *constant term* of P.

In other words,

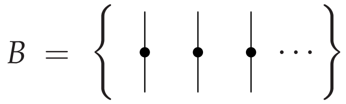
$$P(0) = \{b \in B \mid E_b = 0\} = B \smallsetminus \operatorname{Im} p.$$

linear

1.2.3 Linear functors. Consider now a family given by a bijection $E \xrightarrow{\sim} B$. Since every fibre is a singleton set, the polynomial functor is the *linear functor*

$$X \mapsto \sum_{b \in B} X^1 = B \times X.$$

So in graphical terms, the set B consists entirely of unary operations:



There is a particularly important case of this: the polynomial functor represented by the family $1 \to 1$ is the identity polynomial $X \mapsto X$.

Linear functors in one variable are not terribly exciting (cf. linear algebra in one variable), but linear functors in many variables provide an interesting and useful 'categorified matrix algebra', as we shall see in Section 9.1.

affine

1.2.4 Affine functors. Consider a family given by an injection $E \hookrightarrow B$. There is induced a partition of B into two sets B_0 and B_1: the subset $B_0 \subset B$ consists of the elements with empty fibre, and B_1 consists of the other elements (which by injectivity have a singleton fibre). The elements in B_0 are the nullary operations, pictured like ; the elements in B_1 are the unary operations, pictured like . The functor represented by this family is an *affine functor*, i.e. the sum of a constant functor and a linear functor:

$$X \mapsto \sum_{b \in B_0} X^0 + \sum_{b \in B_1} X^1 \; = \; B_0 \; + \; B_1 \times X.$$

1.2.5 The free-pointed-set functor; the exceptions monad. A particularly important affine functor is

$$X \mapsto 1 + X.$$

It is the functor that freely adds a basepoint to a set. We shall see it is an example of a monad, and we shall need it several times.

In computer science this monad is used to model errors, also called exceptions. Roughly, the idea is something like this: think of a program as a function. Sometimes a program needs to give up, if it comes into a state where is has no instructions on what to do. This is not only for badly written programs — for examples programs that involve interaction with the user, sometimes the user just asks for something meaningless, and the program is at loss. Of course the program should not just crash, it should rather deal with the exceptional situation, for example beep and do nothing and hope the next input from the user makes more sense. The possibility of error means that rather than a function we have a partially defined function. Now a partially defined function with codomain X is the same thing as a well-defined function with codomain $1 + X$, where 1 is now defined as the value of all the undefined points, so the extra value 1 serves as the error value.

Now of course it quickly becomes practical to distinguish between different errors, like error 404 (page not found) and error 504 (timeout), so instead of having just one error, modelled by the set 1, we can have a set E of different exceptions, and look at the monad

$$X \mapsto E + X$$

This is what is called the *exceptions monad* [84]. (This is all a very vague explanation, and we have not at all touched upon where the monad comes in. Hopefully there will be some more substantial explanations in Chapter 6.)

bottom-top

1.2.6 Bottom set and top set. Let $p : E \to B$ represent a polynomial functor P. The bottom set B can be recovered as $B = P(1)$. In a moment we will define the derivative P' of a polynomial functor P, and see that the top set E is naturally identified with $P'(1)$.

1.2.7 Homogeneous polynomials. By definition, the degree of a monomial X^F is the set F. By definition, a polynomial is *homogeneous of degree* F if the representing map is of form $B \times F \to B$. (Or more generally if there are specified bijections between all the fibres of $E \to B$. Let F be one such fibre; since all the other fibres are in canonical bijections with this set, any fibre will do.) Another way of characterising a homogeneous polynomial functor in terms of its representing map $E \to B$ is to say that it is a pullback of a trivial map $F \to 1$ along $B \to 1$.

free-monoid

1.2.8 Fundamental example: the free monoid functor. The *free monoid* $M(X)$ on a set X is the set of all finite words in the alphabet X, including the empty word. (It is a monoid under concatenation of words.) We also write $M(X) = X* = \sum_{n\geq 0} X^n$. The *free monoid endofunctor*

$$\begin{array}{rcl} M : \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & X* = \displaystyle\sum_{n\geq 0} X^n. \end{array}$$

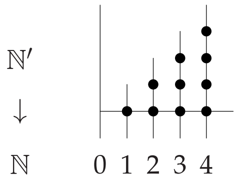
is polynomial: the representing map is the 'universal family'

$$u : \mathbb{N}' \to \mathbb{N}$$

whose fibre over $n \in \mathbb{N}$ is the standard n-element set $\mathbf{n} = \{0, 1, 2, \ldots, n - 1\}$. One way to realise $\mathbb{N}'$ is

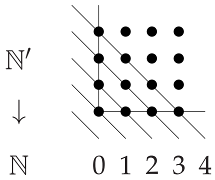
$$\mathbb{N}' = \sum_{n\in\mathbb{N}} \mathbf{n} = \{(n, i) \in \mathbb{N} \times \mathbb{N} \mid i < n\}.$$

Then $\mathbb{N}' \to \mathbb{N}$ is simply the projection $(n, i) \mapsto n$:



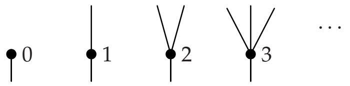
Another way to realise $\mathbb{N}'$ is

$$\begin{array}{rcl} \mathbb{N}' := \mathbb{N} \times \mathbb{N} & \xrightarrow{u} & \mathbb{N} \\ (a, b) & \longmapsto & a + b + 1 \end{array}$$



In fact, if we think of $\mathbb{N}$ as the set of all finite sets, then we might also describe $\mathbb{N}'$ as the set of all pointed finite sets. This is just like in the theory of moduli of curves: the universal family is the space of pointed things. . .

In any case, the elements of the set $\mathbb{N}$, interpreted as operations, cf. the graphical interpretation of 1.1.4, are



Since there is precisely one operation of each arity, we don't really have to label the nodes in each bouquet. . .

(Note that the polynomial $\sum_{n\geq 0} X^n$ is the geometric series, so it is tempting to write

$$M(X) = \frac{1}{1-X},$$

but of course at this point this can at most be a suggestive notation —it cannot be taken literally since it involves division and negative sets! However, in Section 3 we shall introduce negative sets and give some meaning to the expression.)

This functor will play a crucial role. We shall see it is naturally a monad.

1.3 Other viewpoints

1.3.1 Two fundamental isomorphisms. We use all the time:

$$\sum_{b\in B} X = B \times X, \qquad \prod_{e\in E} X = X^E$$

alternative

1.3.2 'Power series'. In usual polynomials we like to collect all monomials of the same type (i.e., the same degree, in the one-variable case) and put a coefficient in front of it to indicate how many there were of it. For polynomial functors, strictly speaking there can never be any repetition among the exponents, because the fibres E_b are all distinct sets. We saw that homogeneous polynomials have the property that there are canonical identifications between the fibres, so in this case we allowed ourselves to write homogeneous polynomials as $B \times X^F$.

In the general case what we can do is to take a cruder approach, contenting ourselves with non-canonical isomorphisms, and simply collect all terms corresponding to sets of the same cardinality. If we assume that $p : E \to B$ is a finite map, i.e., each fibre is a finite set, then we can consider the *classifying map*

kappa

$$\begin{aligned} \kappa : B &\longrightarrow \mathbb{N} \qquad (1.1)\\ b &\longmapsto |E_b| \qquad (1.2) \end{aligned}$$

sending an element b to the cardinality of the corresponding fibre. Putting

$$B_n := \kappa^{-1}(n)$$

we can write B as a sum like this:

$$B = \sum_{n \in \mathbb{N}} B_n$$

and collect terms with isomorphic exponents:

$$\sum_{b \in B} X^{E_b} = \sum_{n \in \mathbb{N}} \sum_{b \in B_n} X^{E_b} \simeq \sum_{n \in \mathbb{N}} \sum_{b \in B_n} X^{\mathbf{n}} = \sum_{n \in \mathbb{N}} B_n \times X^{\mathbf{n}}.$$

(Note that the sign $\simeq$ represents a non-canonical isomorphism.)

This sort of expression,

$$X \mapsto \sum_{n \in \mathbb{N}} B_n \times X^n$$

might be called a power series, for obvious reasons. It is specified completely by giving a sequence of sets $(B_n \mid n \in \mathbb{N})$. But it would be unfortunate terminology if 'power series' would be a more finite notion than 'polynomial', as it is the case here! so we'll refrain from that terminology, and in any case we are not going to exploit this viewpoint very much.

Conversely, to give a functor $X \mapsto \sum_{n \in \mathbb{N}} B_n \times X^n$ is the same thing as giving the sequence $(B_n \mid n \in \mathbb{N})$, and that in turn is just to give an abstract 'classifying map' $\kappa : B \to \mathbb{N}$. Then we can construct a polynomial functor $E \to B$ by taking E to be the pullback:

$$\begin{tikzcd}
E \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N}' \arrow[d, "u"] \\
B \arrow[r] & \mathbb{N}
\end{tikzcd}$$

We could agree to use always the following set as pullback:

$$E = \sum_{b \in B} \mathbb{N}'_{\kappa(b)}.$$

We will come back to these viewpoints in the section on collections and operads.

abstract-preview

1.3.3 More abstract approach. The construction of $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$ from a set map $p : E \to B$ is really a special case of something more general and fundamental, and in some sense also easier to understand. And in any case this is the viewpoint that will lead directly to the many-variable case—we will come back to this viewpoint in Section 8.

Given a set map $p : E \to B$ then there is induced a functor $p^* : \textbf{\textit{Set}}/B \to \textbf{\textit{Set}}/E$ which sends an object $Y \to B$ to the pullback $Y \times_B E \to E$. (Thinking of this as $\textbf{\textit{Set}}^B \to \textbf{\textit{Set}}^E$ then this is just the Yoneda embedding of p, right?) This functor has adjoints on both sides. The left adjoint $p_!$ is easy to describe: it simply sends $Z \to E$ to $Z \to E \to B$. (In the previous paragraph we implicitly computed $\kappa_!$.) The right adjoint p_* is the interesting one just now. It is given by

$$\begin{array}{rcl} p_* : \textbf{\textit{Set}}/E & \longrightarrow & \textbf{\textit{Set}}/B \\ Z & \longmapsto & \displaystyle\sum_{b \in B} \prod_{e \in E_b} Z_e \end{array}$$

where Z_e denotes the fibre over e of $Z \to E$.

Our polynomial functor $X \mapsto \sum_{b \in B} X^{E_b}$ is just a special case of that formula: we start with an abstract set X, and construct a set over E, namely $Z := X \times E$, the pullback of X along the map $s : E \to \mathbf{1}$. Now we apply p_* to get a set over B, and since the map $Z \to E$ is now just the 'trivial fibration' $Z = X \times E \to E$, we can simplify the above formula a little bit, recovering our usual formula:

$$\sum_{b \in B} \prod_{e \in E_b} Z_e = \sum_{b \in B} \prod_{e \in E_b} X = \sum_{b \in B} X^{E_b}.$$

In fact this last set is a set over B. To forget about the B-structure is to apply $t_!$.

1.4 Basic operations on polynomials

Since polynomial functors take values in $\textbf{\textit{Set}}$, and since $\textbf{\textit{Set}}$ is a semiring category, it is clear how to add and multiply polynomial functors. Also, since polynomial functors are endofunctors, it is clear what it should mean to compose them. We shall see now that the results of these operations are again polynomial functors in a canonical way. In fact we shall show how to perform these operations on the level of the representing families.

Polynomial functors are represented by a few sets and maps encoding exponents and coefficients, and we want to describe all operations on polynomial functors in terms of such data.

We now introduce sums and products of polynomials. Soon we shall describe a category of polynomials functors, and then these sums and products will be the categorical sum and product. We treat composition in Subsection 1.5.

1.4.1 Addition of polynomials. Given two polynomial functors P and Q, their sum should be the functor

$$X \mapsto P(X) + Q(X).$$

If P is represented by the family $p : E \to B$ and Q by $q : F \to C$, then it is easy to see that the sum of these two families,

$$p + q : E + F \to B + C,$$

represents the sum of the functors.

Note that this operation is associative (as much as sum is in ***Set***) and that the neutral polynomial functor for addition is the zero functor (represented by the family $0 \to 0$).

product

1.4.2 Multiplication of polynomials. If P and Q are polynomial functors, their product should be the functor

$$X \mapsto P(X) \times Q(X).$$

If P is represented by $p : E \to B$ and Q by $q : F \to C$, let us write out what the product functor does (using distributivity):

$$\begin{aligned} P(X) \times Q(X) &= \Big(\sum_{b \in B} X^{E_b} \Big) \times \Big(\sum_{c \in C} X^{F_c} \Big) \\ &= \sum_{(b,c) \in B \times C} X^{E_b} \times X^{F_c} = \sum_{(b,c) \in B \times C} X^{E_b + F_c}. \end{aligned}$$

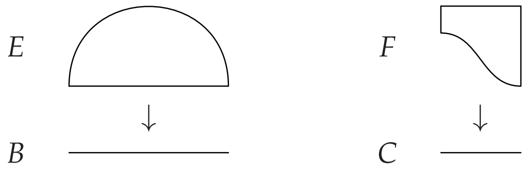
So we see that the base set is $B \times C$ (this is the indexing set for the exponents), and the total space is

$$\sum_{(b,c) \in B \times C} E_b + F_c$$

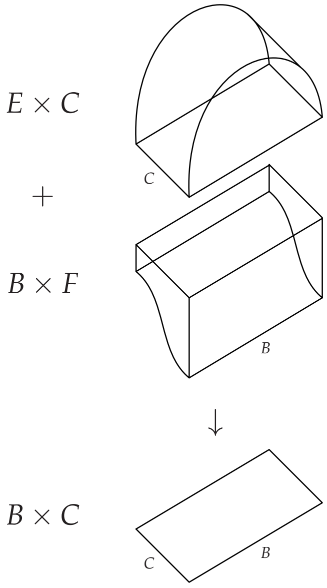
Now this way of writing the total space, just as a sum of all the fibres, always works, but often there is a more synthetic way of describing the set. In this case we find this more concise description of the top space and the map:

$$(E \times C) + (B \times F) \xrightarrow{\left\{\begin{array}{l} p \times \mathrm{id}_C \\ \mathrm{id}_B \times q \end{array}\right.} B \times C$$

Here is a drawing: if P and Q are represented by families drawn like this:



then the drawing of the family representing the product is this:



Note that the multiplication is associative (to the same extent as the product is in ***Set***) and that the neutral polynomial functor is the constant polynomial 1 (represented by the family $0 \to 1$)

Since addition and multiplication are defined in terms of addition and multiplication in ***Set***, it is immediate that the distributive law holds.

We shall shortly introduce the category of polynomial functors —the full subcategory of $\boldsymbol{Fun}(\boldsymbol{Set}, \boldsymbol{Set})$ consisting of polynomial functors. In here, the product we have just described in of course the categorical product (since products are computed pointwise in a functor category).

1.4.3 Scalar multiplication. We have seen (1.1.5) that every set S defines a polynomial functor, namely the constant functor with value S. It is represented by $\emptyset \to S$. We can think of constant functors as scalars. Scalar multiplication is just a special case of the multiplication described above: the scalar multiplication law is this: S times $E \to B$ is equal to $S \times E \to S \times B$.

1.4.4 Elementary-school flashback. If all this looks complicated then try to multiply two good old-fashioned polynomials:

$$\begin{aligned}(x + x^3)(x^8 + 1) &= x \cdot x^8 + x \cdot 1 + x^3 \cdot x^8 + x^3 \cdot 1 \\ &= x^{1+8} + x^{1+0} + x^{3+8} + x^{3+0}\end{aligned}$$

This is a sum indexed by the product of the indexing sets, and for each such indexing pair, the actual exponent is their sum—just as in the general formula.

1.4.5 Sneak preview of Leibniz' rule. In a minute we will introduce the derivative of a polynomial functor. We have already stated that the top set of a representing map $p : E \to B$ can be recovered as $P'(1)$. Of course we want the Leibniz rule to hold:

$$(p \times q)' = (p' \times q) + (p \times q')$$

Hence we can guess what the top set for $(p \times q)$ should be:

$$= (p \times q)'(1) = p'(1) \times q(1) + p(1) \times q'(1) = E \times C + B \times F$$

This is really sort of backwards reasoning, but often mathematics works like this: before we settle on a definition of derivation and products we already require that Leibniz' rule should hold, so if we can't arrange the definitions for this to hold true, then we aren't really interested in the theory. . .

poly-mono

1.4.6 Example. Suppose $p : E \to B$ is any polynomial and $m : F \to 1$ is a monomial. Then in terms of set maps, $p \times m$ is the map $p + (B \times m) : E + (B \times F) \to B$. In terms of the functor it is $X \mapsto \sum_{b \in B} X^{E_b + F}$.

1.5 Composition of polynomials (substitution)

subsec:composition

THIS IS SORT OF BORING TO DO IN DETAIL, UNLESS SOME SIMPLIFICATION SHOWS UP (ONE SUCH SIMPLIFICATION IS THE ABSTRACT APPROACH OF SECTION 8.

1.5.1 Key example: Q is the constant polynomial functor represented by some fixed set S. Then all the operations are nullary, and the composition consists in grafting such dots upon the leaves of the elements of B (the set of operations of P).

1.5.2 Overview. Given set maps $p : E \to B$ and $q : F \to C$, we want to show that the composite of their polynomial functors

$$\mathbf{Set} \xrightarrow{Q} \mathbf{Set} \xrightarrow{P} \mathbf{Set}$$

is again a polynomial functor (as our experience with good-old polynomials suggests). (In a more polynomial language we could say that we substitute Q into P.) Also we'll not be surprised to see that this is a little bit complicated—already for good-old polynomials we know that such a substitution can produce a huge polynomial!

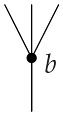
Checking this is just a matter of writing out the formulae, rearrange the terms using distributivity, and see that it is indeed of the correct form, represented by some set map $U \to A$. In Section 8 we will do this from an abstract viewpoint. Here we try to do it by hand.

Now before even trying to expand, we can already say what the base set A is going to be: indeed we know that

$$A = (P \circ Q)(\mathbf{1}) = P(Q(\mathbf{1})) = P(C).$$

This is the set $\sum_{b \in B} C^{E_b}$. So it remains to find the total space U, and describe the projection map $U \to A$.

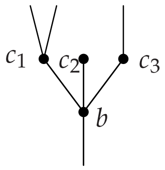
Let us start with a graphical analysis. Recall that an element of B is pictured as a bouquet:



The set of leaves symbolises the fibre E_b. Similarly, an element in E is a bouquet just like this, but with one leaf marked. The map $E \to B$ just forgets this mark.

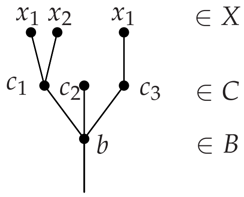
Now look at the composite $P \circ Q$. According to the above argument, the base set of $P \circ Q$ is the set $P(C)$. Its elements are pairs (b, α) where $b \in B$ and $\alpha : E_b \to C$. In other words, pick an element in B (i.e. such a bouquet) and for each leaf pick an element in C. USE THE DESCIPTION OF EVALUATION OF POLYNOMIAL FUNCTORS, DONE IN 1.1.6

IMPORTANT: C is already a set of operations, so in order to keep the dynamical aspect, we just graft the elements of C on top of the elements of B. So $P(C)$ is described in terms of grafting of bouquets: its elements are bouquets of bouquets, or two-level trees, if you wish, where the level-1 node (the root node) is in B and the level-2 nodes are in C: the operations of $P \circ Q$ look like this:



The conditions on the individual bouquets are still in force: at level 1: the edges coming into dot b correspond to the elements in E_b, and on the level 2: for each dot c, the incoming edges are in bijection with F_c. (Since we are in the one-variable case there is no compatibility conditions on which bouquets can be grafted onto which leaves. When we come to the many-variable case, there will some bookkeeping to do.)

When we evaluate on some fixed set X, we get these figures:



(where as an illustration we have let two leaves have the same decorating element $x_1 \in X$).

Now we want to describe U, and if the tree-picture is going to have any value we want the fibre over such a tree to be the set of its leaves. So for a fixed such tree (b, α), what are the leaves? Well, they are the union of all the leaves of the bouquets c, so it is of the form

$$\sum F_c$$

So what are we summing over? Obviously we are summing over E_b, so the end result for the fibre over (b, α) is

$$\sum_{e \in E_b} F_{\alpha(e)}$$

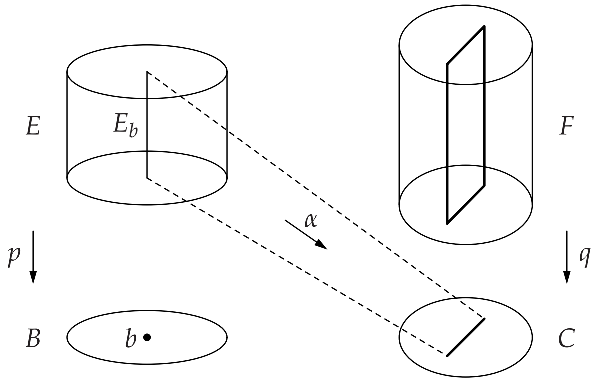
So the global description of U is:

$$\sum_{b \in B} \sum_{\alpha : E_b \to C} \sum_{e \in E_b} F_{\alpha(e)}$$

You can also think: $U_{(b,\alpha)} = \alpha^* F = \{(e, f) \in E_b \times F \mid \alpha(e) = q(f)\}$. So in summary,

$$U = \{(b, \alpha, e, f) \mid b \in B, \alpha : E_b \to C, e \in E_b, f \in F, \alpha(e) = q(f)\}.$$

Here is a picture:



1=generic

1.5.3 Blank. IMPORTANT: $1 = \{\texttt{blank}\}$ is the generic place holder.

1.5.4 Example: composition of linear functors. Recall from 1.2.3 that a linear functor is one represented by a bijection $E \xrightarrow{\sim} B$. We might as well restrict to identity maps. So a linear functor is given by a single set. Now check that the composite of the linear functor given by B with the linear functor given by A is the linear functor given by $B \times A$. This is one-variable linear algebra, and it objectifies the fact that composition of linear maps $\mathbb{R} \to \mathbb{R}$ is just about scalar multiplication. When we come to many-variable linear functors we shall see that matrix multiplication is objectified by the fibre product.

1.5.5 Sneak preview of the chain rule. DO SOMETHING WITH THIS! IT MAKES NO SENSE UNTIL WE HAVE INTRODUCED DIFFERENTIATION! Of course we want the chain rule to hold for composite polynomials:

$$(p \circ q)' = (p' \circ q) \times q'.$$

Now according to 1.2.6, the top set of $p \circ q$ is naturally identified with the bottom set of $(p \circ q)'$. Hence we expect to find

$$U = (p \circ q)'(1) = (p' \circ q)(1) \times q'(1) = p'(C) \times F.$$

P'CxF **1.5.6 Comparison with the direct description.** So what is $P'(C) \times F$?

$$P'(C) \times F = \sum_{e \in E} C^{E_{p(e)} - e} \times F = \sum_{b \in B} \sum_{e \in E_b} C^{E_b - e} \times F.$$

First of all, let's explain how this is a set over $A = P(C) = \sum_{b \in B} C^{E_b}$. We fix $b \in B$ once and for all, so we need to describe a map

$$\sum_{e \in E_b} C^{E_b - e} \times F \longrightarrow C^{E_b}.$$

Now to describe this map it is enough to describe it on each component. So we just need a map $C^{E_b - e} \times F \longrightarrow C^{E_b}$ for each e. So given a punctured map $\alpha' : E_b - e \to C$ (not defined on $e \in E_b$) together with an $f \in F$, how do we extend to a complete map $E_b \to C$? Well, we only need to define the value on e, and the only natural thing to do is to take $q(f)$. So now we have described the map $P'(C) \times F \to P(C)$. This is the map $U \to A$ we are looking for. It remains to see that it agrees with the elementary description given above. To this end, just check out the fibre: over a fixed $b \in B$, $\alpha : E_b \to C$ in $P(C)$, the fibre consists of pairs $\alpha' : E_b - e \to C$ and $f \in F$ for some $e \in E_b$. Such that α extends α', and such that $q(f) = \alpha(e)$. Now the set of choices of an $e \in E_b$ and an extension of α' to the whole of E_b, that's just the set of maps $\gamma : E_b \to C$. But there is the compatibility condition in order to map to α downstairs, namely $\gamma(e) = q(f)$. But this is precisely the primitive description we gave first.

Now let us start from scratch and compute the composite by direct calculation. Let us start with two easy cases.

1.5.7 Substitution of monomials Given sets $m : E \to 1$ and $n : F \to 1$, which we interpret as monomials $m(X) = X^E$ and $n(X) = X^F$, then we can easily compute the composite $m \circ n$ (first apply n, then apply m), so n is inner and m is outer.

$$(m \circ n)(X) = m(n(X)) = m(X^F) = (X^F)^E = X^{F \times E}$$

So the composite is again a monomial, and it just comes from the set map $m \circ n : F \times E \to 1$.

The next case is when instead of the monomial m we have a general polynomial $p : E \to B$. Now

$$(p \circ n)(X) = p(n(X)) = p(X^F) = \sum_{b \in B} (X^F)^{E_b} = \sum_{b \in B} X^{F \times E_b}$$

So the result is the polynomial corresponding to the set map

$$F \times E \to B$$

(Note that we get some strange orders of the factors. This is due to the fact that we wrote composition from right to left. . .)

All this is to say that substitution is linear in the outer argument. Of course this is what we expect from plain-old polynomials, and it is also what we have experienced in the first approach where it was clear that all arguments are fibre-wise for a fixed $b \in B$, which is just another way of phrasing reduction to the case where p is a monomial.

Of course substitution of polynomials is not linear in the inner argument—in general $(x^a + x^b)^e \neq x^{ae} + x^{be}$. Here we get in clinch with distributivity:

We could now treat the general case, but since in fact we have linearity in the outer variable it is enough to treat the case of an outer monomial $m : E \to 1$ and an inner polynomial $q : F \to C$:

$$(m \circ q)(X) = m(q(X)) = m\big(\sum_{c \in C} X^{F_c} \big) = \big(\sum_{c \in C} X^{F_c} \big)^E$$

Now we have to use distributivity to turn this into a polynomial on standard form. Continuing the computation,

$$\big(\sum_{c \in C} X^{F_c} \big)^E = \sum_{\alpha \in C^E} \prod_{e \in E} X^{F_{\alpha(e)}}$$

This is the same as we found initially.

To understand this distributivity argument, let us work out a simple example of plain old polynomials.

1.5.8 Elementary school flashback. Compute $(x^3 + x^5 + x^7)^2$. In the inner polynomial, there are three different exponents, so the indexing set for them is $\mathbf{3} = \{0, 1, 2\}$. Put $f_0 = 3$, $f_1 = 5$, and $f_2 = 7$. So now we can write the inner polynomial as $\sum_{i \in \mathbf{3}} x^{f_i}$.

Now expand by hand:

$$\begin{aligned}(x^3 + x^5 + x^7)^2 \quad = \quad & x^3x^3 + x^3x^5 + x^3x^7 + \\ & x^5x^3 + x^5x^5 + x^5x^7 + \\ & x^7x^3 + x^7x^5 + x^7x^7\end{aligned}$$

There are $9 = 3^2$ terms in the sum, one for each pair of elements in $\mathbf{3}$. Each term is a product of 2 monomials.

To give a pair of elements in $\mathbf{3}$ is just to give a map $\alpha : \mathbf{2} \to \mathbf{3}$, and then the corresponding term in the sum is $\prod\limits_{i \in \mathbf{2}} x^{f_{\alpha(i)}}$. So altogether we have computed the expansion as

$$\sum_{\alpha \in \mathbf{3}^{\mathbf{2}}} \prod_{i \in \mathbf{2}} x^{f_{\alpha(i)}},$$

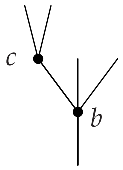
which may look a bit complicated but this is how it is. (Rather, argue the other way around: if you think this formula is complicated, then just write out the 9 terms separately and you are as happy as when you computed the expansion by hand.)

partial

1.5.9 Partial composition. Let us briefly describe another operation, where instead of grafting onto all the leaves we only graft onto one leaf. Recall that the base set $P(C)$ for the polynomial functor $P \circ Q$ is the set of pairs (b, α) where b is an element in B and $\alpha : E_b \to C$ is a map from the fibre, i.e., a decoration of the set of leaves. Now in the new simpler case of partial grafting, we don't need decoration on all leaves of b but only on one of them. So the base is the set of triples (b, e, α) where $b \in B$, and $e \in E_b$ and $\alpha : \{e\} \to C$. Now to give this is just to give $e \in E$, and $c \in C$. So the base is merely

$$C \times E.$$

The picture is simple too:



Now we want to say that the fibre over such a thing is the set

$$\operatorname{leaves}(c) \cup \operatorname{leaves}(b) \smallsetminus \{e\}$$

So this is easy: there are two components: either the leaf is one of the original leaves of the bouquet c—that's F_c. Or it is one of the original leaves of b, but not e. So that's $E_b \smallsetminus e$. In other words, the polynomial functor is nothing but

$$Q \times P'.$$

which spelled out is

$$\begin{tikzcd} F \times E \; + \; C \times (E \times E \smallsetminus \Delta) \arrow[d, "{\begin{cases} q \times 1_E \\ 1_C \times p' \end{cases}}"'] \\ C \times E \end{tikzcd}$$

This is the sum of all possible one-substitutions. It is not like the Stasheff way of specifying the substitution law in terms of several fixed laws $\circ_i$, which only makes sense if there is an ordering. Since we cannot indicate in a global uniform manner a specified element of each fibre, we need to sum over them all.

1.6 Differential calculus

1.6.1 The derivative of a monomial. Given a monomial $P(X) = X^E$, the derivative should be something like $P'(X) = E \times X^{E-1}$. Here we need to give a meaning to the expression $E - 1$: it should mean the set E except for one element, but which element? Well, there are E elements to choose from, and by 'coincidence' there are also E copies of the monomial, so we can just remove each element in turn. Hence a good definition seems to be

$$P'(X) := \sum_{e \in E} X^{E \smallsetminus \{e\}}$$

Note that the derivative of a monomial is no longer a pure monomial: either it has a coefficient, or as we see is better is really a *polynomial*. As such it is represented by a family with base E. The fibre over $e \in E$ is a copy of E with e removed, so the top set of the family is $E \times E$ minus the diagonal, with the first projection as map to E. In conclusion, the derivative of P is represented by the family

$$E \times E \smallsetminus \Delta \to E.$$

We will denote the derivative of $P(X)$ by $DP(X)$ or $P'(X)$ depending on what seems most convenient in the context.

1.6.2 Example. Among the monomials we have the constant $X^0 = 1$, represented by the family $0 \to 1$. Working directly with the definition, we see that the derivative is the empty sum, i.e. the zero polynomial 0. Working instead with the representing families, we find that the derivative is represented by $0 \times 0 \smallsetminus 0 \to 0$, again the zero polynomial.

1.6.3 Derivative of a general polynomial. In general, given a polynomial $P(X) = \sum\limits_{b \in B} X^{E_b}$, represented by $p : E \to B$, we should just use linearity and define

$$P'(X) := \sum_{b \in B} \sum_{e \in E_b} X^{E_b \smallsetminus \{e\}}$$

This functor is clearly polynomial. The base set of the representing family is $\sum_{b \in B} E_b = E$, and the fibre over $e \in E$ is $E_{p(e)} \times E_{p(e)} \smallsetminus \{e\}$. Joining all the fibres we find that the top set of the representing family is

$$E \times_B E \smallsetminus \Delta,$$

and again the map to E is just the first projection, say.

mark-operator

1.6.4 The mark operator. There is another operator which is simpler: namely the one sending P to the polynomial $X \mapsto XP'(X)$. This is the full fibre product (no diagonal removed). If P is represented by $p : E \to B$ then $XP'(X)$ is represented by the top map

$$\begin{tikzcd}
E \times_B E \arrow[r] \arrow[d] & E \arrow[d] \\
E \arrow[r, "p"'] & B
\end{tikzcd}$$

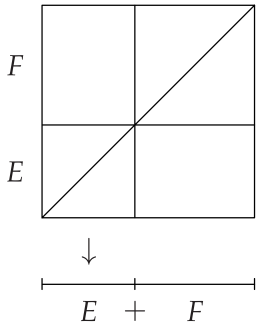
1.6.5 Lemma. *Leibnitz rule holds for differentiation of polynomial functors: given P and Q, then*

$$(P \times Q)' = (P' \times Q) + (P \times Q').$$

Proof. By additivity, it is enough to check the rule when P and Q are monomials, say represented by the sets $p : E \to 1$ and $q : F \to 1$, respectively. By 1.4.2, $P \times Q$ is then represented by $E + F \to 1$, whose derivative is

$$(E+F) \times (E+F) \smallsetminus \Delta \quad \to \quad E + F.$$

You can see it all (including the end of the proof) in this picture:



Since the target set is a sum, the map also splits into a sum: the summand over E is

$$E \times E \smallsetminus \Delta \; + \; E \times F \quad \to \quad E$$

which is precisely the family representing $P' \times Q$. Similarly the F-summand of the family is

$$E \times F \; + \; F \times F \smallsetminus \Delta \quad \to \quad F$$

representing $P \times Q'$. □

1.6.6 Lemma. *The chain rule holds:*

$$(p \circ q)' = (p' \circ q) \times q'.$$

PROOF OF THE CHAIN RULE.

1.6.7 Mac Laurin series (Taylor expansion at zero) Develop this idea: we already know that $P(0)$ is the constant term of P. Similarly, direct computation shows that $P'(0)$ is the set

$$P'(0) = \{e \in E \mid e \text{ is alone in its fibre}\}.$$

Now if P is an affine functor, $P(X) = A + B \times X$, then we can rewrite this as

$$P(X) = P(0) + X \cdot P'(0).$$

So $P(0) = A$ and $P'(0) = B$.

Next, $P''(0)$ is the set of pairs $(e_1, e_2) \in E \times E$ such that $e_1 \neq e_2$ and $p(e_1) = p(e_2)$, and such that there are no other elements in that fibre. In other words the set of pairs of distinct elements in E which constitute a whole fibre.

More generally, $P^{(n)}(0)$ is the set of ordered n-tuples from E that constitute a whole fibre.

This set has a canonical action of $\mathfrak{S}_n$. We denote the quotient set by

$$\frac{P^{(n)}(0)}{n!}.$$

Proposition: Assume all fibres of $p : E \to B$ has cardinality n. Then there exists a (non-canonical) bijection

$$P(X) \simeq \frac{P^{(n)}(0)}{n!} \cdot X^n.$$

1.6.8 IDEA TO LOOK AT. Have a look at this definition of differential quotient: given $f : \mathbb{R} \to \mathbb{R}$, define

$$\begin{array}{rcl} \mathbb{R} \times \mathbb{R} \smallsetminus \Delta & \xrightarrow{d^1 f} & \mathbb{R} \\ (x, y) & \longmapsto & \dfrac{f(x) - f(y)}{x - y} \end{array}$$

This function extends to the whole of $\mathbb{R} \times \mathbb{R}$, and its restriction to the diagonal, $\mathbb{R} \to \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is equal to f'.

(Similarly, define $d^2 f : \mathbb{R} \times \mathbb{R} \times \mathbb{R} \smallsetminus$ some diagonals $\to \mathbb{R}$, $(x, y, z) \mapsto \frac{d^1 f(x,y) - d^1 f(y,z)}{x - z}$ or something. . .

1.7 Properties of polynomial functors

Give only easy arguments here. Refer to 8.6 for fancier and more conceptual proofs in the general case. . .

preserve-pullbacks

1.7.1 Lemma. *A polynomial functor preserves fibre products (and hence monos) (And more generally limits over diagrams with a terminal object). See Part II for such general statements.*

Proof. The statement is a special case of Lemma 8.2.5, and a more conceptual proof is given there. Here we give a more direct ad hoc proof. . .

□

1.7.2 Example. The free monoid functor M does preserve monos (since this takes place degree-wise, so it amounts to the observation that if $X \hookrightarrow Y$ is a mono then $X^E \to Y^E$ is also mono (CHECK THIS OUT)). This should also be the general argument.

We call a set map *finite* if it has finite fibres.

preserve-seq

1.7.3 Lemma. *The polynomial functor preserves filtered colimits if and only if the representing map $E \to B$ has finite fibres.*

Proof. The functor is a sum of representables, so it preserves filtered colimits if and only if each summand does. But each summand is of the form $X \mapsto X^E$, and such preserve filtered colimits if and only if E is finite. □

See also [75].

1.7.4 Example. The free-monoid monad M of 1.2.8: given a directed union like this $A_0 \subset A_1 \subset A_2 \subset \ldots$, then $M(\cup A_i) = \cup M(A_i)$. The first set is the set of all finite words in the total alphabet $\cup A_i$. Now any such word has only a finite number of letters so it does belong to A_i for some sufficiently big i.

eth

Chapter 2

Categories of polynomial functors in one variable

Sec:morphisms

There are two useful categories of polynomial functors: one is the full subcategory of $\boldsymbol{Fun}(\boldsymbol{Set}, \boldsymbol{Set})$ consisting of polynomial functors. That is, all natural transformations are allowed as morphisms between polynomial functors. We denote it $\boldsymbol{Set}[X]$. [POSSIBLY THIS IS BAD NOTATION BECAUSE IT INVOLVES POLYNOMIAL FUNCTORS WITH OPERATIONS OF INFINITE ARITIES. THAT'S NOT BAD IN ITSELF, BUT THE NOTATION $\boldsymbol{Set}[X]$ MIGHT SUGGEST THAT WE ARE TALKING ABOUT SOMETHING FREELY GENERATED BY BINARY SUMS AND BINARY PRODUCTS. BUT THAT WOULD GIVE ONLY FINITARY POLYNOMIAL FUNCTORS. . .]

The other alternative is more restrictive: we only allow cartesian natural transformations. We denote it ***Poly***.

The first has better categorical properties, and it works as a categorification of the usual polynomial semirings. However the general morphisms are a bit tricky to handle in terms of the representing maps $E \to B$, and some important constructions do not work so well in this generality. The cartesian ones behave very well in term of representing families, and they are important in the applications to operads and tree structures.

2.1 The category $\mathbf{Set}[X]$ of polynomial functors

The main result is that every natural transformation between polynomial functors factors as a representable one followed by a cartesian. This is to say that the functor

$$\begin{array}{rcl} \mathbf{Set}[X] & \longrightarrow & \mathbf{Set} \\ P & \longmapsto & P(1) \end{array}$$

is a Grothendieck fibration. Of course we are going to introduce the terms used.

2.1.1 Categories of polynomial functors. Polynomial functors form a category denoted $\mathbf{Set}[X]$ by taking the arrows to be all natural transformation between them. [SEE REMARKS AT THE START OF THIS CHAPTER]

A main goal in this section is to describe natural transformations of polynomial functors in terms of their representing families.

2.1.2 Notation. Up to now we have often drawn diagrams of polynomial functors vertically, in order to better visualise the idea of fibre. Now we shall turn the pictures 90 degrees and picture maps $E \to B$ horizontally. This is more practical, and prepares the way for the multi-variable case. . .

2.1.3 Preliminary heuristic discussion. Let P be a polynomial functor represented by $p : E \to B$, and let Q be represented by $q : F \to C$. The question we ask is how to describe the natural transformations $P \Rightarrow Q$ in terms of the representing maps?

A first guess might be that natural transformations should correspond to commutative squares

$$\begin{tikzcd} E \arrow[r, "p"] \arrow[d] & B \arrow[d] \\ F \arrow[r, "q"'] & C. \end{tikzcd}$$

This is not true however—in general such a square does not induce a natural transformation $P \Rightarrow Q$. We shall see in 2.1.7 that if the square is a pullback square then it works: there is induced a natural transformation, but not every natural transformation arises in this way.

The problem with commutative squares occurs already in the case of monomials: Given monomials p and q and a commutative square

$$\begin{tikzcd}
E \arrow[r, "p"] \arrow[d] & 1 \arrow[d] \\
F \arrow[r, "q"'] & 1
\end{tikzcd}$$

there is no way in general to construct a natural map $X^E \to X^F$ from it. The only natural map goes in the other direction—in fact by the Yoneda lemma for

$$\begin{array}{rcl} \boldsymbol{Set}^{\mathrm{op}} & \longrightarrow & \boldsymbol{Fun}(\boldsymbol{Set},\boldsymbol{Set}) \\ E & \longmapsto & [X \mapsto X^E], \end{array}$$

there is a bijection between natural transformations $X^E \to X^F$ and maps $F \to E$. Let us take note of this observation in a slightly more general set-up:

B=B **2.1.4 Representable transformations.** Given a commutative square

$$\begin{tikzcd}
E \arrow[r, "p"] & B \arrow[d, equal] \\
F \arrow[u, "\sigma"] \arrow[r, "q"'] & B
\end{tikzcd}$$

then there is induced a natural transformation $\sigma : P \Rightarrow Q$:

$$\sum_{b\in B} X^{E_b} \to \sum_{b\in B} X^{F_b}$$

which sends the b-summand into the b-summand—i.e. it is a B-map. We'll call such transformations *representable*.

To see the construction formally, note that the triangle amounts to a map $q \to p$ in the category ***Set***$/B$. That's the same as a family of maps $F_b \to E_b$. In other words, the set of all such triangles is the set

$$\prod_{b\in B} \mathrm{Hom}_{\boldsymbol{Set}}(F_b, E_b).$$

But by the contravariant Yoneda lemma, this set is the same as

$$\prod_{b\in B} \mathrm{Nat}(X^{E_b}, X^{F_b})$$

(where abusively we denote the functor $X \mapsto X^{E_b}$ by X^{E_b}). So for each $b \in B$ we have a natural transformation $X^{E_b} \to X^{F_b}$. Now we can sum over all those to get

$$\sum_{b\in B} X^{E_b} \to \sum_{b\in B} X^{F_b}$$

In fact, the argument shows also that every natural transformation $\sum_{b\in B} X^{E_b} \to \sum_{b\in B} X^{F_b}$ which respects the b-summands arises uniquely in this way. The natural transformation can also be characterised as those for which the 1-component

$$B = P(1) \to Q(1) = B$$

is the identity map. So we have characterised those natural transformations in terms of easy triangle diagrams.

There is an obvious way to stack two such diagrams. It is easy to see that

2.1.5 Lemma. *Stacking of such diagrams corresponds precisely to composition of the associated natural transformations.*

Back to the idea of squares: while general commutative squares do not induce natural transformations, the situation is much better with pullback squares, as already mentioned.

Cartesian morphisms

cartesian

Definition. If $\mathscr{D}$ and $\mathscr{C}$ are categories with fibre products, given two functors $F, G : \mathscr{D} \to \mathscr{C}$, a natural transformation $u : F \Rightarrow G$ is *cartesian* if for every arrow $X \to Y$ in $\mathscr{D}$, the naturality square

$$\begin{tikzcd}
F(X) \arrow[r] \arrow[d] & G(X) \arrow[d] \\
F(Y) \arrow[r] & G(Y)
\end{tikzcd}$$

is cartesian.

F(1) **2.1.6 Remark.** Suppose $\mathscr{D}$ has a terminal object 1, and denote by $u_X : X \to 1$ the unique map from an object X to 1.

If $\theta : F \Rightarrow G : \mathscr{D} \to \mathscr{C}$ is a cartesian natural transformation, then the cartesian naturality square

$$\begin{tikzcd}
F(X) \arrow[r, "F(u_X)"] \arrow[d, "\theta_X"'] & F(1) \arrow[d, "\theta_1"] \\
G(X) \arrow[r, "G(u_X)"'] & G(1)
\end{tikzcd}$$

shows that F is determined by its value on 1 and its relation to G. Indeed, the cartesian square gives a natural isomorphism

$$F(X) = G(X) \times_{G(1)} F(1).$$

cartNatTrans **2.1.7 Proposition.** *Given two polynomial functors P and Q, represented by $p : E \to B$ and $q : F \to C$ respectively. Then the cartesian natural transformations from Q to P correspond precisely to cartesian squares*

$$\begin{tikzcd}
E \arrow[r, "p"] \arrow[d, "\overline{\alpha}"'] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "\alpha"] \\
F \arrow[r, "q"'] & C
\end{tikzcd}$$

Proof. Given the pullback square, we construct canonically a natural cartesian transformation $P \Rightarrow Q$: we must define for each set X a map

$$\sum_{b \in B} X^{E_b} \to \sum_{c \in C} X^{F_c}.$$

Since the square is cartesian, for each $b \in B$ there is a canonical isomorphism $E_b \xrightarrow{\sim} F_{\alpha(b)}$. The inverse of this isomorphism induces an isomorphism $X^{E_b} \xrightarrow{\sim} X^{F_{\alpha(b)}}$. Since we have such a map for each $b \in B$ we have defined the wanted map. To check naturality with respect to a map $\varphi : X \to Y$, it is enough to check for the c-summand: here it is obvious that this square commutes and it is even cartesian:

$$\begin{tikzcd}
X^{E_b} \arrow[r] \arrow[d] & X^{F_{\alpha(b)}} \arrow[d] \\
Y^{E_b} \arrow[r] & Y^{F_{\alpha(b)}}
\end{tikzcd}$$

—the horizontal maps are isomorphisms! In fact it follows readily from this, that also the general naturality squares are cartesian: when setting all the sum signs, each fibre for the horizontal maps are locally isomorphisms: for each $c \in C$, the inverse image of X^{F_c} is $\sum_{b\in\alpha^{-1}(c)} X^{E_b}$. And each E_b is isomorphic to F_c. Hence, locally at c, the map is just the second projection from the product $\alpha^{-1}(c) \times X^{F_c}$.

A more down-to-earth explanation: to give a cartesian morphism is to give a map $\alpha : B \to C$ together with a bijection $E_b \xrightarrow{\sim} F_{\alpha(b)}$ for each $b \in B$. So it's about setting up bijections between the exponent sets.

Conversely, given a cartesian natural transformation, we construct a pullback square. In fact we can do better, as in Proposition 2.5.1. □

2.1.8 Lemma. *Composition of cartesian natural transformation (between polynomial functors) corresponds precisely to pasting of pullback squares between the representing maps.*

kappa-not-unique **2.1.9 Example.** (Cf. Example 1.2.8.) For any finite map $p : E \to B$, the classifying map (1.1) extends to a cartesian square

$$\begin{tikzcd}
E \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "\kappa"] \\
\mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

and hence a cartesian morphism from P to the free monoid functor M. So the universal property can be restated by saying that every finite polynomial functor has a natural transformation to M. NOTE HOWEVER that this natural transformation is rarely unique, because although κ is uniquely defined, the map in the left-hand side of the diagram is not! (The natural transformation to M is unique if and only if P is affine, i.e. when $E \to B$ is mono.)

2.1.10 Example. Another obvious example of such a pullback square is

$$\begin{tikzcd}
E \times_B E \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d] \\
E \arrow[r, "p"'] & B
\end{tikzcd}$$

The two horizontal maps represent polynomial functors and the square represents a cartesian natural transformation between them. We saw in 1.6.4 that the top functor is $X \cdot P'(X)$.

2.1.11 Putting together the pieces. Now we have two ways of inducing natural transformations: the representable ones and the cartesian ones. Of course we can compose these two kinds to obtain new natural transformations. For example we can compose a representable natural transformation with a cartesian one: We get natural transformations represented by diagrams of the form

$$\begin{tikzcd}
E \arrow[r, "p"] & B \arrow[d, equal] \\
Z \arrow[u, "\sigma"] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d] \\
F \arrow[r, "q"'] & C
\end{tikzcd} \tag{2.1}$$

generalmorphism

We could also compose in the opposite order—first a cartesian transformation and then a representable one, like this:

$$\begin{tikzcd}
E \arrow[r, "p"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d] \\
W \arrow[r] & C \arrow[d, equal] \\
F \arrow[u] \arrow[r, "q"'] & C
\end{tikzcd}$$

Now the crucial remark is that this last composite natural transformation can be realised by a diagram of the first kind. The argument goes like this: simply consider the pullback $Z = F \times_C B$. Now this set has maps to W and B, so by the universal property of the pullback E, there is then induced a unique map $\sigma : Z \to E$, and we are back to the standard situation. Of course it remains to check that the natural transformation induced by this cartesian-representable composite is the same as the one induced by the original representable-cartesian one. This is routine.

So any composite of cartesian and representable natural transformations are of the form of diagram 2.1.

Now we are ready to formulate the main result in this section:

2.1.12 Proposition. *Every natural transformation between polynomial functors factors as a representable transformation followed by a cartesian transformation, and this factorisation is essentially unique.*

In other words:

2.1.13 Proposition. *Representable natural transformations and cartesian natural transformations form a factorisation system in the category* $\mathbf{Set}[X]$.

This operation of changing a cartesian-representable composite into a representable-cartesian one describes precisely how the factorisation system works.

2.1.14 Factoring a polynomial transformation. We will no show how a general polynomial transformation can be factored. Let there be given a natural transformation $s : P \Rightarrow Q$, i.e. a natural family

$$\sum_{b\in B} X^{E_b} \longrightarrow \sum_{c\in C} X^{F_c}.$$

The first set is naturally over B and the second is naturally over C, and the first remark is that each summand maps into a summand. This is just to take the 1-component of the natural transformation: this gives us the map $B \to C$. The naturality diagram reads

$$\begin{tikzcd}
\sum_{b\in B} X^{E_b} \arrow[r] \arrow[d] & \sum_{c\in C} X^{F_c} \arrow[d] \\
B \arrow[r] & C.
\end{tikzcd}$$

which shows that the map splits into summands.

So now we have

$$\begin{tikzcd}
E \arrow[r] & B \arrow[d] \\
F \arrow[r] & C
\end{tikzcd}$$

In summary, for each b we have a map $X^{E_b} \to X^{F_{\varphi(b)}}$, and altogether a map

$$\sum_{b \in B} X^{E_b} \to \sum_{b \in B} X^{F_{\varphi(b)}}.$$

Here the right-hand side is nothing but the polynomial functor U defined by the map

$$Z := F \times_C B \to B$$

so we have found a natural transformation $P \Rightarrow U$. By the Yoneda lemma (for ***Set***$/B$), this is represented by some B-map $U \to E$, i.e. a diagram

$$\begin{tikzcd}
E \arrow[r, "p"] & B \\
Z \arrow[u] \arrow[r] & B. \arrow[u, equal]
\end{tikzcd}$$

On the other hand, there is a cartesian natural transformation $U \Rightarrow Q$, defined by the square

$$\begin{tikzcd}
Z \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d] \\
F \arrow[r] & C
\end{tikzcd}$$

Hence we have factored our natural transformation into $P \Rightarrow U \Rightarrow Q$.

2.1.15 As a Grothendieck fibration. Consider the category of arrows in ***Set***, denoted ***Set***$^{\mathbf{2}}$. The objects are arrows $a : x \to y$, and the morphisms are commutative squares

$$\begin{tikzcd}
x \arrow[r] \arrow[d, "a"'] & x' \arrow[d, "a'"] \\
y \arrow[r, "f"'] & y'
\end{tikzcd}$$

Consider the functor ***Set***$^{\mathbf{2}} \to$ ***Set*** which returns the target of an arrow (and the bottom arrow of such a diagram). This functor is a Grothendieck fibration: the cartesian morphisms are precisely the pullback squares. This means that every morphism in $\boldsymbol{Set}^{\mathbf{2}}$ factors (uniquely) as a vertical morphism followed by a cartesian one. The vertical morphisms are squares

$$\begin{tikzcd}
x \arrow[r] \arrow[d] & x \arrow[d] \\
y \arrow[r, equal] & y'
\end{tikzcd}$$

Hence we see a very strong similarity between $\boldsymbol{Set}[X]$ and $\boldsymbol{Set}^{\mathbf{2}}$. In fact, $\boldsymbol{Set}[X]$ is obtained from $\boldsymbol{Set}^{\mathbf{2}}$ by reversing the direction of all vertical arrows.

2.1.16 Theorem. *The functor*

$$\begin{array}{rcl} \boldsymbol{Set}[X] & \longrightarrow & \boldsymbol{Set} \\ P & \longmapsto & P(1) \end{array}$$

is a Grothendieck fibration.

2.1.17 More advanced remark. The general notion of morphism corresponds to allowing 'repetition and omission' (which makes sense only in cartesian contexts(?), repetition is like using the diagonal map, and omission is like projection. . .) This is like allowing any sort of set map between the fibres of f and the fibres of g. To restrict to cartesian morphisms is to allow only bijections between the fibres of f and g. That is: no repetition, no omission: every element must be used precisely once. . .

2.2 Sums, products

We now describe how to take sums and products of morphisms. It is clear that these constructions are functorial, since they are just sums and products in a functor category. However, we insist on spelling out how the constructions work in terms of representing diagrams.

2.2.1 Naturality of multiplication. Given cartesian morphisms $\alpha_1 : p_1 \Rightarrow q_1$ and $\alpha_2 : p_2 \Rightarrow q_2$, then there is a cartesian morphism

$$\alpha_1 \times \alpha_2 : p_1 \times p_2 \Rightarrow q_1 \times q_2.$$

Concretely, given pullback squares

$$\begin{tikzcd}
E_1 \arrow[r, "p_1"] \arrow[d, "\overline{\alpha}_1"'] & B_1 \arrow[d, "\alpha_1"] & E_2 \arrow[r, "p_2"] \arrow[d, "\overline{\alpha}_2"'] & B_2 \arrow[d, "\alpha_2"] \\
F_1 \arrow[r, "q_1"'] & C_1 & F_2 \arrow[r, "q_2"'] & C_2
\end{tikzcd}$$

Then we can form the square

$$\begin{tikzcd}
E_1 \times B_2 + B_1 \times E_2 \arrow[r, "\langle p_1 \times B_2, B_1 \times p_2 \rangle"] \arrow[d, "\overline{\alpha}_1 \times \alpha_2 + \alpha_1 \times \overline{\alpha}_2"'] & B_1 \times B_2 \arrow[d, "\alpha_1 \times \alpha_2"] \\
F_1 \times C_2 + C_1 \times F_2 \arrow[r, "\langle q_1 \times C_2, C_1 \times q_2 \rangle"'] & C_1 \times C_2
\end{tikzcd}$$

It is straightforward to check that this square is again a pullback square.

Question to check: given a pair of more general morphisms, is their product then again a morphism?

2.2.2 Multiplication with P**.** We already argued that multiplication with P is functorial with respect to cartesian transformations. We now check that it is also functorial with respect to representable transformations. Given a representable transformation $\theta : Q_1 \Rightarrow Q_2$. It is given by a diagram

$$\begin{tikzcd}
F_1 \arrow[r] & C \arrow[d, equal] \\
F_2 \arrow[r] \arrow[u] & C
\end{tikzcd}$$

Multiply each of the functors with $E \to B$, and see that there is a natural map induced between them:

$$\begin{tikzcd}
F_1 \times B + C \times E \arrow[r] & B \times C \arrow[d, equal] \\
F_2 \times B + C \times E \arrow[r] \arrow[u] & B \times C
\end{tikzcd}$$

The map is obvious. Hence we have shown that multiplication with a fixed P is functorial both with respect to cartesian and with respect to representable transformations. Hence by the factorisation result, multiplication with P is functorial on all of $\boldsymbol{Set}[X]$.

multX **2.2.3 Multiplication by a monomial.** Multiplication by a monomial X^n is particularly interesting, because since a monomial is represented by a map $n \to 1$, multiplication with it does not change the base. In other words, the endofunctor on $\boldsymbol{Set}[X]$ given by multiplication with X^n respects fibres for the Grothendieck fibration. For example, multiplication with X sends $E \to B$ to

$$B + E \to B$$

It makes every member of the family one element bigger.

So multiplication with X^n induces an endofunctor on each fibre of the fibration $\boldsymbol{Set}[X] \to \boldsymbol{Set}$. Each fibre, say over C, is equivalent to the opposite of $\boldsymbol{Set}/C$. So multiplication-by-X^n defines an endofunctor on $\boldsymbol{Set}/C$. It sends $E \to B$ to $(n \times B) + E \to B$, as we already know.

2.3 Algebra of polynomial functors: categorification and Burnside semirings

Now that we have a distributive category $\boldsymbol{FinSet}[X]$ of polynomial functors we can finally investigate to what extent polynomial functors are a categorification of the polynomial semiring $\mathbb{N}[X]$. There are two levels of interpretation: a good-old polynomial can be regarded either as a formal expression, i.e. a certain configuration of coefficients and exponents, or it can be regarded as a function from $\mathbb{N}$ to $\mathbb{N}$. One shows that two polynomials define the same function if and only if they are the same formal expression, i.e. their coefficients and exponents agree.

The same is true for polynomial functors:

2.3.1 Theorem. *Two polynomial functors are isomorphic as functors if and only* *if their representing set maps are isomorphic (i.e we have a commutative square*

$$\begin{tikzcd} E' \arrow[r] \arrow[d, "\simeq"'] & B' \arrow[d, "\simeq"] \\ E \arrow[r, "\simeq"] & B \end{tikzcd}$$

(which is of course automatically cartesian)).

Proof. In fact we already proved more: let $\theta : P' \Rightarrow P$ be a natural isomorphism. We know this particular natural transformation is represented uniquely by a square as claimed. □

2.3.2 Categories of polynomial functors. Polynomial functors form a category denoted $\textbf{\textit{Set}}[X]$ by taking the arrows to be all natural transformation between them. We see that $\textbf{\textit{Set}}[X]$ is a distributive category. WHAT DO WE MEAN BY DISTRIBUTIVE? FINITE DISTRIBUTIVE?

2.3.3 Finite polynomial functors. Call a polynomial functor *finite* if the representing family consists of two finite sets; this means that all exponents and coefficients are finite. Let $\textbf{\textit{FinSet}}[X]$ denote the full subcategory of finite polynomial functors.

Now for the categorification result, analogous to 0.0.1.

N[X]-Burnside **2.3.4 Theorem.** $\mathbb{N}[X]$ *is the Burnside semiring of the category of finite polynomial functors.*

This extends the usual categorification of the natural numbers:

$$\begin{tikzcd} \textbf{\textit{FinSet}} \arrow[r, hook] \arrow[d] & \textbf{\textit{FinSet}}[X] \arrow[d] \\ \mathbb{N} \arrow[r, hook] & \mathbb{N}[X] \end{tikzcd}$$

Note that by the above results on preservation of all operations under isomorphisms (and sometimes under more general functors), it is immediate that this categorification accounts for sums, products, substitution, and differentiation.

The proof follows readily from the alternative description of finite polynomial given in 1.3.2: every finite polynomial functor is isomorphic to a functor of the form

$$X \mapsto \sum_{n \in \mathbb{N}} B_n \times X^n$$

with only finitely many B_n non-empty. Clearly, these 'normal-form' polynomials correspond precisely to good-old natural-number polynomials.

semiringhomo

2.3.5 Semiring category. ***FinSet*** is a distributive category, and it sits as a full coreflective subcategory in ***FinSet***$[X]$. This functor is a distributive functor (i.e. preserved sums and products).

The functor

$$\begin{array}{rcl} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set}[X] \\ S & \longmapsto & [0 \to S] \end{array}$$

sending a set S to the constant functor $X \mapsto S$ is a distributive functor, making ***Set***$[X]$ into a ***Set***-algebra.

2.3.6 Lemma. *Every natural transformation out of a constant functor is cartesian.*

Proof. A constant functor is represented by $\emptyset \to S$, and when a representable map out of it would amount to a map to $\emptyset$. But such a map must be the identity map of $\emptyset$. Hence the representable part of the transformation is invertible.

□

P(0)-adj

2.3.7 Lemma. *The inclusion* ***Set*** $\to$ ***Set***$[X]$ *has a right adjoint, given by returning the constant part of a polynomial (i.e. evaluation at* 0*).*

Proof. Natural transformations (cartesian by the previous lemma) from the constant functor $\emptyset \to S$ to $E \to B$ are given by specifying a map $S \to B$, and the elements in B hit by this map must be nullary, by cartesianness. But $P(0)$ is precisely the set of nullary operations. This establishes the asserted bijection

$$\boldsymbol{Set}(S, P(0)) = \boldsymbol{Set}[X](S, P)$$

□

Note in particular that evaluation at 0 is right adjoint to both the inclusion $\boldsymbol{Set} \to \boldsymbol{Set}[X]$ and to the inclusion $\boldsymbol{Set} \to \boldsymbol{Poly}$ (cartesian fragment).

2.3.8 Theorem. *$\boldsymbol{FinSet}[X]$ is the free **FinSet**-algebra on one generator.*

This should mean: first we define what a ***FinSet***-module is: it is a category $\mathscr{C}$ with finite sums and equipped with an action $\boldsymbol{Set} \times \mathscr{C} \to \mathscr{C}$. Now define a ***FinSet***-algebra to be a monoid in the category of ***FinSet***-modules.

The subtlety is that we have two candidates for the free thing: $\boldsymbol{FinSet}[X]$ and also the semiring of formal expressions $\sum_{n \in \mathbb{N}} b_n X^n$. Depending on the notion of monoid and also depending on the notion of action, we get different results: if we take some definition like formal iterated binary products of objects in ***FinSet*** with object X, then possibly we get the latter version. However, we should allow products indexed over arbitrary finite sets, and this will lead to $\boldsymbol{FinSet}[X]$, it seems. The proof still remains to be worked out in detail.

Similarly, ***Set*** is a distributive category and it injects into the distributive category $\boldsymbol{Set}[X]$.

2.3.9 Theorem. *$\boldsymbol{Set}[X]$ is the free **Set**-algebra on one generator.*

THIS IS PROBABLY FALSE WITHOUT SOME MORE CARE WITH INFINITIES. 'FREE ALGEBRA' WILL OFTEN MEAN SOMETHING GENERATED BY FINITELY MANY OPERATIONS, BUT IT SEEMS THAT $\boldsymbol{Set}[X]$ CONTAINS ALSO INFINITE OPERATIONS. . .

2.3.10 Experiment. Just as the result that $\mathbb{N}$ classifies finite sets can be stated more interestingly as a solution to a moduli problem, try to do the same for polynomial functors. Define what a family of polynomial functors is, over a base that can be any set. We should try to encode the fibres of such families as finite set maps. In other words, for each element $j \in J$ in the base set, we should have a finite set map $E_j \to B_j$. In other words, this is to have set maps

$$\begin{tikzcd}
E \arrow[d] \\
B \arrow[d] \\
J
\end{tikzcd}$$

such that for each fixed $j \in J$, the sets E_j and B_j are finite. This in turn is just to have a J-tuple of polynomial functors. It is also clear what an isomorphism of such families is: just an isomorphism on each level.

Then figure out what the universal family should be, the base should be $\mathbb{N}[X]$. So just describe the fibre over some arbitrary polynomial, say $3X^2 + 4$. The corresponding $E_j \to B_j$ should have a 7-element set as B, and over three of the points the fibre should be 2, and over four of the points the fibre should be empty.

We might try to do this in a linear algebra fashion: since $\mathbb{N}[X]$ as an $\mathbb{N}$-module is spanned by $\mathbb{N}$ itself (the monomials, 1, X, X^2, ...), we might try first to describe the fibres over these monomials. The fibre over X^k is just the set map $k \to 1$. The fibre over $a_k X^k$ is just the set map $a_k \times k \to a_k$.

Perhaps we can arrive at a nice global description in the style of those universal families $\mathbb{N}' \to \mathbb{N}$...

2.3.11 Other things to work out — routine. We have the category of commutative semirings. Now we also have the 2-category of distributive categories: the arrows are the functors that preserve finite sums and finite products. The 2-cells are the natural transformations. Just as $\mathbb{N}$ is the initial object in the category of commutative semirings, so ***FinSet*** should be initial in the 2-category of distributive categories. However, this is rather 2-initial: it means that for any distributive category $\mathscr{D}$, the hom cat ***Distr***$(\boldsymbol{FinSet}, \mathscr{D})$ is contractible. In other words, up to equivalence, there is only one sum-product preserving functor to $\mathscr{D}$. It is perhaps better first to consider a skeleton for ***FinSet***, namely the category of finite cardinals (often denoted Φ by me).

2.4 Composition

Here we should only treat horizontal composition of 2-cells if it can be done in a reasonably elementary fashion, i.e. without building up big diagrams which would fit more naturally into the many-variable setting.

The outcome is a monoidal structure on each of ***Set***$[X]$ and ***Poly***.

2.5 The subcategory *Poly*: only cartesian natural transformations

FtoP **2.5.1 Proposition.** *If $P : \boldsymbol{Set} \to \boldsymbol{Set}$ is a polynomial functor represented by $p : E \to B$, and $F : \boldsymbol{Set} \to \boldsymbol{Set}$ is any functor with a cartesian natural transformation $\theta : F \Rightarrow P$, then F is also polynomial (i.e. isomorphic to one such). More* *precisely, F is represented by $E \times_B F(1) \longrightarrow F(1)$, and θ is represented by*

$$\begin{tikzcd}
E \times_B F(1) \arrow[r] \arrow[d] & F(1) \arrow[d, "\theta_1"] \\
E \arrow[r] & B = P(1)
\end{tikzcd}$$

Proof. The polynomial functor represented by $E \times_B F(1) \longrightarrow F(1)$ has the same value on 1 as F, and since they both have a cartesian natural transformation to P they are naturally isomorphic by Observation 2.1.6. □

circcart **2.5.2 KEY REMARK.** *The horizontal composition of two cartesian natural transformations (between polynomial functors) is again cartesian.* This is a consequence of the fact that polynomial functors preserve pullbacks.

2.5.3 Terminology. We call a category *discrete* if the only arrows are the identity arrows. We say that a category is *rigid* if for any two objects there is at most one isomorphism between them. This is equivalent to saying that each object has a trivial group of automorphisms. For groupoids, the two notions coincide up to equivalence, in the sense that any rigid groupoid is equivalent to a discrete one. Maybe we should rather reserve the word discrete for rigid groupoids. . .

2.5.4 Some elementary remarks on set maps and bijections. It is a triviality that if two linearly ordered finite sets are isomorphic (as linearly ordered sets) then the isomorphism is unique. An ordering is just a fixed bijection with some finite ordinal $\mathbf{n}$, so the remark is that the category ***FinBij*** $/\mathbf{n}$ is rigid: between any two sets there is at most one bijection over $\mathbf{n}$. Now orders are not the only way of fixing things. For any set B, the category ***Bij*** $/B$ is rigid.

Key observation to put somewhere explicitly: To say that a square of sets

$$\begin{tikzcd}
E' \arrow[r, "p'"'] \arrow[d, "\psi"'] & B' \arrow[d, "\phi"] \\
E \arrow[r, "p"'] & B
\end{tikzcd}$$

is cartesian is to say that for each $b' \in B'$, the map ψ restricts to a bijection between the fibres $E'_{b'}$ and $E_{\phi(b')}$.

Note crucially that giving the cartesian square *specifies* those bijections. But just from the maps ϕ and p we cannot know which bijection: different choices of the pullback give different bijections. Choosing a pullback amounts to choosing some bijections. It is crucial that we choose and fix a specific pullback.

So ψ is characterised as being fibrewise a bijection. Note that since ϕ is not necessarily surjective, it can happen that some fibre of p is ridiculously large and does not correspond to anything up in E'. The fibres over points in B not in the image are not part of any bijection.

Note that if p' is mono, that means that each fibre $E_{b'}$ is empty or singleton. In this case (and only in this case) is the map ψ determined completely by the map ϕ.

unique-pullback

2.5.5 Remark. We have seen by example that in a pullback square

$$\begin{tikzcd}
F_1 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & C_1 \arrow[d, "\phi"] \\
F_2 \arrow[r] & C_2
\end{tikzcd}$$

there may be many different maps $F_1 \to F_2$ making the diagram commutative. But if both $F_1 \to C_1$ and $F_2 \to C_2$ have a cartesian morphism to some fixed $E \to B$, and ϕ is required to commute with this, then the extension to $F_1 \to F_2$ is unique. Indeed, fibrewise over $b \in B$ the map has to commute with the bijections $(F_1 \to E)_b$ and $(F_2 \to E)_b$, hence is unique.

We shall have a look at the slice categories of ***Poly***.

2.5.6 Proposition. *For a fixed polynomial functor P represented by $E \to B$, the slice category **Poly**/P is naturally equivalent to **Set**/B.*

2.5.7 Remark. If we understand by polynomial functor any functor isomorphic to one given by one of those diagrams, then the functor should be described as $Q/P \mapsto Q1/P1 \in$ ***Set***$/P1$.

Proof. We have the obvious functor $\Phi : \textbf{\textit{Poly}}/P \to \textbf{\textit{Set}}/B$ which to a cartesian square

$$\begin{tikzcd}
W \arrow[r] \arrow[d] & V \arrow[d] \\
E \arrow[r] & B
\end{tikzcd}$$

associates the map $V \to B$. This functor is canonically given. In the other direction we can associate to any map $V \to B$ the pullback square

$$\begin{tikzcd}
E \times_B V \arrow[r] \arrow[d] & V \arrow[d] \\
E \arrow[r] & B
\end{tikzcd}$$

However, for this to make sense we need to choose pullbacks. It might not be strictly functorial but only a pseudo-functor in some suitable 2-categorical setting. . . Assuming we have this, the universal property of the pullback ensures that the two functors define an equivalence of categories. But at least this converse 'construction' makes it clear that the first functor Φ is surjective on objects. To see that it is fully faithful: that's precisely the preceding remark: maps between pullback squares over a fixed map are completely determined by the codomain component ϕ. □

Remark: in particular, all slices of ***Poly*** are toposes. ***Poly*** itself is not since it does not have a terminal object. Later we investigate pullbacks and products. It seems that ***Poly*** has many features in common with the category of topological spaces and etale maps. [The product of two spaces X and Y in this category is the space of germs of etale maps from X to Y (which is canonically isomorphic to the space of germs of etale maps from Y to X). In particular, the product is the empty space in many cases.]

Products in *Poly*

The category ***Poly*** of polynomial functors and cartesian natural transformations have some strange products!

THEY ARE A BIT DEGENERATE. THEY ARE DESCRIBED IN SOME HANDWRITTEN NOTES.

2.5.8 Point-wise products. The pointwise product, the one treated in 1.4.2, is not the categorical product in ***Poly***. Nevertheless, it is still well-defined as a functor on ***Poly***: to establish this we just have to show that the pointwise product of two cartesian natural transformations is again a cartesian natural transformation. In other words, given cartesian natural transformations $\theta_1 : P_1 \Rightarrow Q_1$ and $\theta_2 : P_2 \Rightarrow Q_2$, then the natural transformation $\theta_1 \cdot \theta_2 : P_1 \cdot P_2 \Rightarrow Q_1 \cdot Q_2$ is again cartesian. We can check this directly on the representing families: if P_i is represented by $E_i \to B_i$ and Q_i is represented by $F_i \to C_i$, then we just need to check that the square

$$\begin{tikzcd}
E_1 \times B_2 + B_1 \times E_2 \arrow[r] \arrow[d] & B_1 \times B_2 \arrow[d] \\
F_1 \times C_2 + C_1 \times F_2 \arrow[r] & C_1 \times C_2
\end{tikzcd}$$

is a pullback. We should be able to check this summand-wise: we first check that

$$\begin{tikzcd}
E_1 \times B_2 \arrow[r] \arrow[d] & B_1 \times B_2 \arrow[d] \\
F_1 \times C_2 \arrow[r] & C_1 \times C_2
\end{tikzcd}$$

is a pullback, and afterwards we check the other summand... But this should just amount to saying that the product of two pullback squares is again a pullback square.

So we have an extra monoidal structure on ***Poly***.

2.5.9 Multiplication with X. There is a particularly important case of point-wise multiplication, namely the functor

$$\begin{array}{rcl} \boldsymbol{Poly} & \longrightarrow & \boldsymbol{Poly} \\ P & \longmapsto & X \cdot P \end{array}$$

sending $\sum_{b \in B} X^{E_b}$ to $\sum_{b \in B} X \times X^{E_b} = \sum_{b \in B} X^{1+E_b}$. The representing family of $X \cdot P$ is

$$B + E \to B$$

It makes every fibre one element bigger, and we saw in 2.2.3 how it is related to the free-pointed-set monad $X \mapsto 1 + X$.

Poly_1=Lin

2.5.10 Linear functors. The full subcategory of ***Poly*** consisting of the linear functors is naturally isomorphic to ***Poly***$/\mathsf{Id}$. (In particular ***Lin*** has a terminal object, namely the identity functor.)

The inclusion ***Lin*** $\to$ ***Poly*** has a right adjoint, given by taking product with Id. This works only with this strange categorical product that we have in ***Poly***.

This is a general fact about forgetful functor from slice categories of a category with products.

Differentiation in *Poly*

2.5.11 OBSERVATION! Differentiation is not functorial with respect to general morphisms! Indeed, given a general morphism

$$\begin{tikzcd}
E \arrow[r, "p"] & B \\
Z \arrow[u, "\sigma"] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[u, equal] \arrow[d] \\
F \arrow[r, "q"'] & C
\end{tikzcd}$$

if there were a natural transformation between the derivatives, it would involve a map on bases $E \to F$, and there is no way we can get this map.

2.5.12 Lemma. *Taking derivative is functorial for cartesian morphisms.*

Proof. Given a cartesian natural transformation

$$\begin{tikzcd}
E \arrow[r, "p"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "\varphi"] \\
F \arrow[r, "q"'] & C
\end{tikzcd}$$

pull back the whole cartesian square

$$\begin{tikzcd}
F \times_C F \arrow[r] \arrow[d] & F \arrow[d] \\
F \arrow[r] & C
\end{tikzcd}$$

along φ to get

$$\begin{tikzcd}
E \times_B E \arrow[r] \arrow[d] & E \arrow[r] \arrow[d] & B \arrow[d] \\
F \times_C F \arrow[r] & F \arrow[r] & C
\end{tikzcd}$$

Since the right-hand square and the big composite square are cartesian by construction, the left-hand square is cartesian.

Now if we remove the diagonals, we are just diminishing the cardinality by one in each fibre, both upstairs and downstairs, so we still have a bijection on fibres. GIVE A MORE FORMAL ARGUMENT HERE. □

THIS IS THE APPROPRIATE POINT TO STATE AND PROVE THAT DIFFERENTIATION IS RIGHT ADJOINT TO MULTIPLYING WITH X.

Chapter 3

Aside: Polynomial functors and negative sets

Sec:negative

In this section we look at the possibilities for getting some sort of negative coefficients and exponents for out polynomial functors. The theory of negative sets is a whole subject on its own, and this is not the place to give the detail—and I don't know so much about it. But there are some very funny calculations to do...

3.1 Negative sets

We have seen that $\mathbb{N}$ is the Burnside semiring of ***FinSet*** and that $\mathbb{N}[X]$ is the Burnside semiring of ***FinSet***$[X]$. Can we find a distributive category $\mathscr{D}$ whose Burnside semiring is $\mathbb{Z}$? This question was posed and studied by Schanuel [95], to whom is due all the results of this section. The presentation owes a lot to John Baez [10].

It turns out fairly quickly that there can be no such distributive category, so it is necessary to rephrase the question.

3.1.1 Lemma. *In any distributive category, if* $X_0 + X_1 \simeq 0$ *then also* $X_0 \simeq X_1 \simeq 0$.

Proof. Given an isomorphism $\varphi : X_0 + X_1 \to 0$, let φ_i be defined by the diagram

$$\begin{tikzcd}
X_0 \arrow[r, hook, "i_0"] \arrow[dr, "\varphi_0"'] & X_0 + X_1 \arrow[d, "\varphi"] & X_1 \arrow[l, hook', "i_1"'] \arrow[dl, "\varphi_1"] \\
 & 0 &
\end{tikzcd}$$

But in a distributive category, every initial object is strict, in the sense that any arrow into it is invertible. Hence φ_0 and φ_1 are invertible. □

3.1.2 Corollary. *The Burnside semiring of a distributive category is never a ring.*

(Except if the distributive category is trivial, in which case also its Burnside semiring is trivial, and hence a ring. . .)

Of course that's a serious blow to the project, but Schanuel [95] observed that we can ask for slightly less. If we can't find a distributive category with an object X such that $X + 1 \simeq 0$, then we might at least be able to find X such that $X + 1 \simeq 0$ modulo something.

The following heuristic discussion is borrowed from Baez [10].

3.1.3 Islands and bridges. A finite set is a bunch of isolated islands,

$$\bullet \quad \bullet \quad \bullet$$

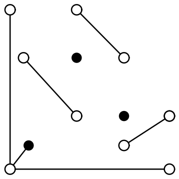
So what is a negative island? A bridge ∘—∘ is a negative island: if you place it between two islands you get just one island! According to this principle, setting two bridges between two islands should give zero islands, which you could also explain by saying that a lake is a negative island, and a pair of bridges between two islands is the same as one island with a lake. Now if you join the two bridges. i.e. fill the lake between them, then you are left with one island again, so a bridge-between-bridges counts as a negative bridge, i.e. counts as 1. With this alternating behaviour it begins to look like the Euler characteristic: set

$$\chi(\bullet) = 1, \qquad \chi(\circ\!\!-\!\!\circ) = -1$$

and demand compatibility with sums and products.

This version of the Euler characteristic, which is called the *Euler measure*, agrees with the ordinary topological Euler characteristic for compact spaces, defined as $\sum_{i=0}^{\infty} h^i(X)$, but in general is defined using cohomology with compact support, assuming this sum converges, which is the case for most nice spaces. One class of spaces where the Euler measure is well-defined, and not too difficult to treat, are the polyhedral sets:

3.1.4 Polyhedral sets. A *polyhedral set* is a subset of $\mathbb{R}^n$ (for any n) obtained from half-spaces by taking intersections, unions, and complements. A *polyhedral map* between polyhedral sets $A \subset \mathbb{R}^m$ and $B \subset \mathbb{R}^n$ is a map whose graph is a polyhedral set (in $\mathbb{R}^{m+n}$). This means that the map may well be discontinuous, but that it is linear on each of finitely many pieces that make up the domain A. Here is an example of the graph of a polyhedral map from the open interval to itself



(In fact you observe that this is an invertible polyhedral map.)

Let $\mathscr{P}$ denote the category of polyhedral sets and maps. It is easy to verify that this is a distributive category: the sum is the disjoint union, so you need some convention of how to embed realise this in some $\mathbb{R}^k$. For example, given $A \subset \mathbb{R}^m$ and $B \subset \mathbb{R}^n$, define $A + B \subset \mathbb{R}^{m+n+1}$ to be

$$A \times \{\mathbf{0}\} \times \{1\} \ \cup \ \{\mathbf{0}\} \times B \times \{2\}$$

or some other convention. The product is just the cartesian product, as polyhedral set in $\mathbb{R}^{m+n}$.

We now restrict attention to the full subcategory $\mathscr{P}_0$ of *bounded polyhedral sets*, i.e. those polyhedral sets that fit into a ball in some $\mathbb{R}^n$.

A *pure d-cell* is a polyhedral set isomorphic to the product of d copies of the open unit interval I. The pure 0-cells are just points. It is a fact that every bounded polyhedral set admits a pure cellularisation, i.e. can be written as a disjoint union

$$A = \sum_{d=0}^{\infty} a_d I^d$$

of a_d copies of pure d-cells I^d.

With $\bullet$ and $\circ\!\!-\!\!\circ$ we can generate all bounded polyhedral sets. The open interval I plays a key role, as a model for -1. The crucial observation is that it satisfies the equation

$$I \simeq I + 1 + I,$$

which is just to say that we can cut the interval into two intervals with a 'gluing point' in the middle:

$$\circ\!\!-\!\!\circ = \circ\!\!-\!\!\circ + \bullet + \circ\!\!-\!\!\circ$$

This relation shows that $\circ\!\!-\!\!\circ$ is as good a model for -1 as we can hope for: while $0 = \circ\!\!-\!\!\circ + 1$ is impossible, at least we have $\circ\!\!-\!\!\circ = \circ\!\!-\!\!\circ + \bullet + \circ\!\!-\!\!\circ$, so we can also say that we have found a -1 module additive cancellation.

3.1.5 The Euler measure. CLEAN UP THE NEXT COUPLE OF PARAGRAPH

Intuitively, the Euler measure should work just like the Euler characteristic, except that you should use only open cells. We want it to count I^d as $(-1)^d$, and of course it should be finitely additive and multiplicative. The existence of such a measure goes a long way back in history and is known as the

3.1.6 Hadwiger-Lenz lemma. (Euler and Rota) *There exists a function $\chi : \mathscr{P}_0 \to \mathbb{Z}$ which is finitely additive and multiplicative, and with $\chi(\bullet) = 1$, and $\chi(I) = -1$.*

The Euler measure is also called the combinatorial Euler characteristic, and Baez calls it the Euler-Schanuel characteristic, but in fact the function fits naturally into geometric measure theory.

The Euler measure can also be characterised in terms of cellularisations: define if $A = \sum_{d=0}^{\infty} a_d I^d$, define $\chi(A) = \sum_{d=0}^{\infty} (-1)^d a_d$, but then you have to prove that it is independent of choice of cellularisation, and that once again is a bit similar to proving that the map from $\mathbb{E}$ is injective. . .

The construction of Schanuel, and also Rota, to prove the Hadwiger-Lenz lemma, uses Euler integration. Note first that polyhedral maps have polyhedral fibres: indeed, the fibre can be seen as a linear section of the graph, which is polyhedral by definition of polyhedral map. Next, observe that a $\mathbb{Z}$-valued polyhedral map must have finite image.

Now suppose we are given a measure χ, and a polyhedral map $f : A \to \mathbb{Z}$. The Euler integral of f with respect to χ is defined as

$$\int_A f \, d\chi := \sum_{n \in \mathbb{Z}} n \, \chi(A_n),$$

where $A_n := f^{-1}(n)$. Now construct the Euler measure inductively: first define a measure χ_1 for all polyhedral sets of dimension $-\infty$, 0 and 1, by linearity and the initial requirements $\chi_1(\circ\!\!-\!\!\circ) = -1$, and so on. Now let E be a 2-dimensional polyhedral set, and let $p : E \to B$ denote its projection to the first coordinate axis. Then we have a sort of classifying map

$$\begin{array}{rcl} \kappa_p : B & \longrightarrow & \mathbb{Z} \\ b & \longmapsto & \chi_1(E_b). \end{array}$$

Check that this map is polyhedral, but at least it has polyhedral fibres. . .

Now define the second Euler measure as

$$\chi_2(E) := \int_B \kappa_p \, d\chi_1.$$

And check the axioms

PERHAPS WE SHOULD NOT GO INTO THIS HERE, BUT JUST STATE AS A THEOREM THAT SUCH AN EULER MEASURE EXISTS. . .

3.1.7 Remark. If X is a finite set of points, then $\chi(X) = \#X$.

If X is compact, then the Euler measure coincides with the topological Euler characteristic. But in general it does not, and in particular the Euler measure is not a homotopy invariant. The basic example is of course the open interval which has Euler measure -1, although it is homotopically equivalent to a point.

Now you can go on an use polyhedral sets and their Euler measure to do tricks with negative sets, but it is worthwhile to understand how the theory relates with Burnside semirings and such.

3.1.8 Theorem. (Schanuel and also Rota.) *The Burnside semiring of $\mathscr{P}_0$ is*

$$\mathbb{N}[T]/(T{+}1{+}T \sim T)$$

which we denote by $\mathbb{E}$.

By the above discussion it is clear that there is a surjection of semirings from $\mathbb{E}$ to the Burnside semiring of $\mathscr{P}_0$: just send T to the class of the open interval. The difficult part is to show the map is injective. To this end, we need some algebraic remarks. The idea is that $\circ\!\!-\!\!\!-\!\!\circ$ fails to be a true -1 because of lack of additive cancellation. . .

3.1.9 Additive cancellation. The semiring $\mathbb{N}$ has additive cancellation, and so has $\mathbb{Z}$:

$$n + x = n + y \quad \Rightarrow \quad x = y.$$

Let ***½Ring***$^{\pm}$ denote the full subcategory of semirings having additive cancellation. The inclusion functor i : ***½Ring***$^{\pm} \hookrightarrow$ ***½Ring*** has a left adjoint a which to any semiring R associates the quotient semiring $R/\sim$, where $\sim$ is the congruence defined by

$$x \sim y \Leftrightarrow \exists r[r + x = r + y]$$

The counit for the adjunction is (for each semiring R) simply the quotient map $R \to R/\sim$. Every homomorphism of semirings $R \to T$, where T has additive cancellation, factors uniquely through the quotient map $R \to R/\sim$. This quotient map is called the *Euler measure* on a semiring, and we denote it

$$\chi : R \to R/\sim$$

We also talk about the Euler measure on a distributive category, which of course is just the Euler measure on its Burnside semiring. We shall see shortly that this gives the Euler measure on $\mathscr{P}_0$.

Note in any case that this abstract Euler measure generalises cardinality, because for the distributive category ***FinSet***, the Burnside semiring already has additive cancellation, so a is the identity in this case.

3.1.10 Lemma. *The additive-cancellatification of $\mathbb{E}$ is $\mathbb{Z}$:*

$$a(\mathbb{E}) = \mathbb{Z},$$

and the Euler measure is $T \mapsto -1$.

This is easy to see: we already have the relation $2T + 1 = T$, and if we impose additive cancellation this gives $T + 1 = 0$, so T goes to -1.

Now notice that there is a well-defined notion of degree in $\mathbb{E}$, since the relation we divide out by equates polynomials of the same degree. This defines a homomorphism of semirings to the semiring

$$\mathbb{D} := (\mathbb{N} \cup \{-\infty\}, \max, +)$$

where addition is max and multiplication is the natural-number sum. (Note that the degree of a sum of polynomials is the max of their degrees, and that the degree of a product is the sum of the degrees.) This semiring is called the *dimension semiring* by Schanuel, and in other contexts it is called the tropical semiring.

With a little work one can find in fact that:

3.1.11 Lemma. *Two elements in $\mathbb{E}$ are equal if and only if they have the same Euler measure and the same degree. In other words,*

$$\mathbb{E} \to \mathbb{Z} \times \mathbb{D}$$

in injective.

It is easy to characterise the image of this map: it is nearly surjective, except in dimension $-\infty$ and 0: for $d = -\infty$ necessarily we have $\chi = 0$, and for $d = 0$ necessarily we have $\chi > 0$. Otherwise all values are attained.

With these preparations we see that defining a map from the Burnside semiring of $\mathscr{P}_0$ to $\mathbb{E}$, is equivalent to defining maps $\chi : \mathscr{P}_0 \to \mathbb{Z}$ and $d : \mathscr{P}_0 \to \mathbb{D}$. The first is the Euler measure, as defined geometrically, and the second is the expected geometric notion of dimension: the *dimension* of a polyhedral set A is the largest d for which there exists an injective polyhedral map $I^d \to A$, and we set $d(\emptyset) = -\infty$.

Given these two geometrically defined invariants, we have therefore defined a map from the Burnside of $\mathscr{P}_0$ to $\mathbb{E}$, and from the constructions it is easy to see that it is inverse to the 'presentation map' initially given.

So as a corollary of these results we find that:

3.1.12 Corollary. *Two bounded polyhedral sets are isomorphic if and only if they have the same dimension and the same Euler measure.*

This may look strange at first, since the dimension only depends on the highest dimension component. So for example, a solid square plus a point is isomorphic to the sum of two squares.

But note that the whole notion of component is sort of out of place, because there is no continuity involved anywhere.

You might say that the reason the result is true, is that the highest dimensional component gives the necessary space to absorb and rearrange the lower dimensional pieces. For example, starting with a solid square plus a point, the solid square can split off a half open square •—∘ × closed interval, and this new piece in turn is isomorphic to a triangle minus a vertex. And this vertex we had extra from the beginning, so we get a solid triangle, which in turn is isomorphic to a solid square. . .

3.1.13 Other models. There are other geometric categories for which the Euler measure is Z-valued. Schanuel outlines also the examples of the category of semi-algebraic sets, where instead of generating the geometric objects by affine inequalities, we use polynomial inequalities. This category turns out to have $\mathbb{E}$ as Burnside semiring. In this case there is no need to restrict to bounded sets, because the open interval is semi-algebraically isomorphic to the open real half-line ($t \mapsto t^{-1} - 1$). Another example is the category of finitely subanalytic sets, again with Burnside semiring $\mathbb{E}$.

It is also interesting to consider *constructible sets*: i.e. the boolean closure of the set of algebraic varieties in $\mathbb{C}^n$. This time the Burnside semiring is big and complicated, and its Euler measure will not seem to be $\mathbb{Z}$-valued. However, by dividing out with yet another relation, which is roughly to force the Fubini theorem to hold, one does get $\mathbb{E}$ again. This relation says that if the fibres of a map $A \to B$ have the same measure, then the measure of B is the product of the measure of A with the measure of the fibre. (Note that this relation holds already for the Euler measure for the three categories mentioned above.)

I DON'T REALLY UNDERSTAND THIS. PLEASE READ SCHANUEL, TO SEE IF YOU UNDERSTAND IT BETTER.

3.2 The geometric series revisited

We saw that the universal family of finite sets parametrises the free monoid monad $M(X) = \sum_{n=0}^{\infty} X^n$, and we suggested to write it as $\frac{1}{1-X}$. Now that we have negative sets at our disposal, let us make a computation to support this suggestion. Just like the standard first-year calculus proof, let us multiply M with the polynomial functor $H(X) = 1 - X$, and see if we get something like 1.

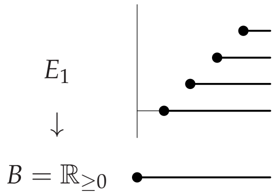
Let us first describe the representing family of H: it is the disjoint union of the trivial family $0 \to 1$ (which represents the constant polynomial functor 1), and the family ○—○ → ○—○ which represents the polynomial $-X$ (it has exponent 1 and coefficient ○—○, hence the family has base ○—○ and singleton fibres). Another heuristic argument: evaluate at 1 to get the base (that's -1), and evaluate the derivative at 1 to get the top space (that's also -1). Putting the two pieces together we see that H is represented by this family:

$$\begin{tikzcd}
\text{○—○} \arrow[d] \\
\text{●—○}
\end{tikzcd}$$

Now compute $M \times H$ by applying the formula for multiplication of polynomial functors 1.4.2. The base space B of the product $M \times H$ is the product of the bases, so we get

$$B = \mathbb{N} \times \text{●—○} = \sum_{i \in \mathbb{N}} \text{●—○} = \mathbb{R}_{\geq 0}$$

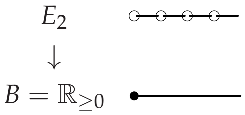
Now let us compute the top space: according to the product rule, it is the sum $E = E_1 + E_2$ of two components: the first is $E_1 = \mathbb{N}' \times$ ●—○ with the projection $u \times \mathrm{id}$ down to $\mathbb{N} \times$ ●—○ $= \mathbb{R}_{\geq 0} = B$. Here is a picture of that map:



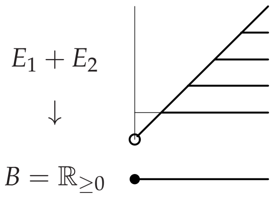
The top space can be described as $E_1 = \{(x, i) \in \mathbb{R}_{\geq 0} \times \mathbb{N} \mid i < x\}$, and then the map to the base $B = \mathbb{R}_{\geq 0}$ is just the projection. We see that over the first ●—○ $\subset \mathbb{R}_{\geq 0}$, the fibre is empty, over the next such piece it is singleton, and so on.

Now the other part of the top space is the base of the first factor times the top space of the second. That space is just $E_2 = \mathbb{N} \times$ ○—○ $= \sum_{i \in \mathbb{N}}$ ○—○ $=$

$\mathbb{R}_{\geq 0} \smallsetminus \mathbb{N}$, the positive reals minus the integer points. The map to the common base $B = \mathbb{R}_{\geq 0}$ is just the inclusion (corresponding to the fact that the family representing R is just the inclusion of the open interval into the half-open):



So add this piece to the picture from before, to get something like



Hence there is a single empty fibre (over $0 \in \mathbb{R}_{\geq 0}$). Over the first ○—● the fibre is singleton, over the next half-open interval the fibre is of cardinality 2, and so on. So altogether the polynomial functor is

$$(M \cdot H)(X) = 1 + \sum_{n>0} \bullet\!\!-\!\!\circ\, X^n$$

This is not precisely the constant polynomial functor as we might have hoped for, but it is so modulo ●—○, and that was in fact all we could reasonably hope for.

Hence we have given a sort of bijective proof of an analytical identity, by showing that two polynomial functors are isomorphic, up to dimension reduction. Recall that two polyhedral sets are isomorphic if and only if they have the same dimension and the same Euler measure. We have shown that the identity holds up to dimension.

3.2.1 Remark. Note that the base set of the universal family is not a finite set and that it is not a polyhedral set either. Also, the identification we made, $\mathbb{N} \times$ ●—○ $\simeq \mathbb{R}_{\geq 0}$ is not an isomorphism of polyhedral sets, it is merely a bijection of abstract sets.

3.3 Moduli of punctured Riemann spheres

in this subsection we consider an example which is interesting since it combines differentiation of polynomial functors with interpretation in terms of negative sets.

3.3.1 Moduli of punctured Riemann spheres. Let $B^{(n)}$ denote the moduli space of Riemann spheres with $3+n$ labelled punctures, modulo puncture preserving holomorphic isomorphism, and we let $r^{(n)} : E^{(n)} \to B^{(n)}$ denote the universal family. (The standard notation for $B^{(n)}$ in algebraic geometry is $M_{0,3+n}$, where the subscript indicates genus 0.) Given a 3-punctured Riemann sphere, there is a unique isomorphism with $\mathbb{CP}^1 \smallsetminus \{0,1,\infty\}$, so clearly $B^{(0)} = 1$, and the total space of the universal family is just $E^{(0)} := \mathbb{CP}^1 \smallsetminus \{0,1,\infty\}$. In view of this, we can fix automorphisms once and for all, and simply take $B^{(n)}$ to be the space of n labelled punctures in $\mathbb{CP}^1 \smallsetminus \{0,1,\infty\}$, without having to mod out by anything else. Hence simply

$$B^{(1)} = E^{(0)} = \mathbb{CP}^1 \smallsetminus \{0,1,\infty\},$$

and the universal family is the fibred product minus the diagonal:

$$E^{(1)} = B^{(1)} \times B^{(1)} \smallsetminus \Delta,$$

and in general,

$$E^{(n)} = \underbrace{B^{(1)} \times \cdots \times B^{(1)}}_{n+1 \text{ factors}} \smallsetminus \text{ all diagonals.}$$

An alternative description, which is the one we shall use, is

$$E^{(n)} = B^{(n)} \times_{B^{(n-1)}} B^{(n)} \smallsetminus \Delta;$$

here $r^{(n)} : E^{(n)} \to B^{(n)}$ is the projection onto the first factor.

In summary, we've got a tower of maps

$$\begin{tikzcd}
\cdots \arrow[r] & B^{(3)} & & & \\
& E^{(2)} \arrow[u, equal] \arrow[r, "r^{(2)}"] & B^{(2)} & & \\
& & E^{(1)} \arrow[u, equal] \arrow[r, "r^{(1)}"] & B^{(1)} & \\
& & & E^{(0)} \arrow[u, equal] \arrow[r, "r^{(0)}"] & B^{(0)} \\
& & & & 1 \arrow[u, equal]
\end{tikzcd}$$

realconf

3.3.2 Remark. In fact, once we have fixed $E^{(0)}$ as $\mathbb{CP}^1 \smallsetminus \{0, 1, \infty\}$ and only consider configurations of labelled punctured in here, without dividing out any further, we might as well take $E^{(0)}$ to be any other space or set, and the construction is the same: $B^{(n)}$ becomes the set of configurations of labelled punctures in $E^{(0)}$. One interesting case is to take $E^{(0)}$ to be an open interval in $\mathbb{R}$.

3.3.3 The polynomial functors corresponding to $r^{(n)}$. Let $R^{(0)}$ denote the polynomial functor represented by $r^{(0)}$. It is just the monomial

$$R^{(0)}(X) = X^{E^{(0)}}.$$

Now the easy observation is that all the others are precisely the iterated derivatives of this one! The description of $r^{(n)}$ as a projection from a fibre product minus the diagonal is exactly the definition of derivative. Hence:

$$D^n R^{(0)} = R^{(n)}.$$

3.3.4 Interpretation in terms of negative sets. We only defined the Euler measure for bounded polyhedral sets. For the unbounded case, you need also to assign the value -1 to the open real half-line. If X denotes the open interval as before, and if we let Y denote the open half line, we have the relation $Y \simeq X + 1 + Y$. Schanuel shows that this extra relation gives the Burnside semiring of $\mathscr{P}$. Now we should only notice that

$$E^{(0)} = \mathbb{CP}^1 \smallsetminus \{0, 1, \infty\} \text{ has Euler measure} - 1.$$

Intuitively, since the $\mathbb{CP}^1$ is compact its Euler measure equals is Euler characteristic, which is 2, and removing three points leaves us with Euler measure -1. More formally, we are talking about the complex plane minus two points, and it is easy to cut in into pieces: we get 4 pieces of type $\mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$, four pieces of type $\mathbb{R}_{\geq 0}$, and one piece ∘—∘. Hence by additivity,

$$\chi(E^{(0)}) = 4 - 4 - 1 = -1.$$

So we can interpret

$$R^{(0)}(X) = X^{E^{(0)}} = X^{-1},$$

and similarly

$$\begin{aligned} R^{(n)}(X) &= D^n \, R^{(0)}(X) \\ &= D^n \, X^{-1} \; = \; -n!(-X)^{-n-1}. \end{aligned}$$

To do the next one by hand, notice that the fibres of $r^{(1)} : E^{(1)} \to B^{(1)}$ are all Riemann spheres with four punctures, temporarily denoted F, hence of Euler measure $\chi(F) = -2$. Hence

$$\begin{aligned} R^{(1)}(X) &= \sum_{b \in B^{(1)}} X^{E_b^{(1)}} \; \simeq \; B^{(1)} \times X^F \\ &= -X^{-2} \; = \; D \, X^{-1}. \end{aligned}$$

3.3.5 The easier, bounded version. As noticed in 3.3.2, we might redefine all $E^{(n)}$ and $B^{(n)}$ to refer to configurations of labelled punctures in the open interval. The negative-set interpretations are the same since $\chi(\circ\!\!-\!\!\circ) = -1$, and in fact it is an simpler model for those derivatives of X^{-1}, because we can stay in the category of bounded polyhedral sets. The choice of punctured Riemann spheres was chosen mainly because the audience were algebraic geometers. However there are a couple of other reasons for the choice: one is that $\mathbb{CP}^1 \smallsetminus \{0, 1, \infty\}$ can also be viewed as $\mathbb{C} \smallsetminus \{0, 1\}$, which is the generator (the -1) in the category of constructible sets (after quotienting by the Fubini relation)

Chapter 4

Algebras

4.1 Initial algebras, least fixpoints

4.1.1 P**-algebras.** Let $\mathscr{C}$ be a category and consider an endofunctor $P : \mathscr{C} \to \mathscr{C}$. A P-*algebra* is a pair (A, a) consisting of an object A of $\mathscr{C}$ together with an arrow $a : P(A) \to A$. A morphism of P-algebras from (A, a) to (B, b) is just an arrow $f : A \to B$ such that this square commutes:

$$\begin{tikzcd}
P(A) \arrow[r, "P(f)"] \arrow[d, "a"'] & P(B) \arrow[d, "b"] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

This defines the category P***-alg*** of P-algebras. The P-algebras are also called *Lambek algebras* for P.

4.1.2 Remark. Note that there are no axioms imposed on the structure map $a : P(A) \to A$. Now if P happens to be a monad, there is another notion of algebra, Eilenberg-Moore algebras: in that case the structure map is required to satisfy the associative and unit axioms, amounting to demanding (A, a) to be a right P-module (well, a left module, in the current backwards right-to-left notation for composition):

$$\begin{tikzcd}
PPA \arrow[r, "Pa"] \arrow[d, "\mu A"'] & PA \arrow[d, "a"'] & A \arrow[l, "\eta A"'] \arrow[dl, "="] \\
PA \arrow[r, "a"'] & A &
\end{tikzcd}$$

In other words, in the case where P is a monad, an Eilenberg-Moore algebra is a Lambek algebra satisfying two extra axioms.

When P is a monad, or if it just has a natural transformation $\mu : PP \Rightarrow P$, then $P(A)$ is automatically a P-algebra: the structure map is $\mu A : PPA \to PA$. If P is a monad, then this is also an Eilenberg-Moore algebra.

The following result (due to Lambek [68]) is very very useful, yet very easy to prove.

4.1.3 Lemma. *(Lambek's fixpoint lemma.) If (I, η) is an initial object of the category of P-algebras, then η is an isomorphism.*

Proof. Consider the P-algebra $(P(I), P(\eta))$ and the diagram

$$\begin{tikzcd}
P(I) \arrow[r, "P(u)"] \arrow[d, "\eta"'] & P(P(I)) \arrow[r, "P(\eta)"] \arrow[d, "P(\eta)"'] & P(I) \arrow[d, "i"] \\
I \arrow[r, "u"'] & P(I) \arrow[r, "\eta"'] & I.
\end{tikzcd}$$

The right-hand square is obviously commutative, saying that η is a morphism of P-algebras from $(P(I), P(\eta))$ to (I, η). In the left-hand square, u is the unique P-algebra morphism to $(P(I), P(\eta))$ from (I, η); hence by definition the left-hand square commutes too. Now since (I, η) is initial, we conclude that $\eta \circ u$ is the identity arrow of I. On the other hand, $u \circ \eta = P(\eta) \circ P(u) = P(\eta \circ u) = P(\mathrm{id}_I) = \mathrm{id}_{P(I)}$; hence u is the inverse to η. □

4.1.4 Least fixpoints. An object X with an isomorphism $P(X) \xrightarrow{\sim} X$ is called a *fixpoint*, so the lemma says that an initial algebra is a fixpoint, via the forgetful functor from P-***alg*** to $\mathscr{C}$. In fact it is the *least fixpoint* in the sense that it does not contain any proper subalgebras. Indeed, if I is an initial P-algebra, and $X \subset I$ is a subalgebra (more precisely, we are given a monomorphic P-algebra homomorphism $f : X \hookrightarrow I$), then since I is initial, there is a unique P-algebra homomorphism $u : I \to X$, and the composite map $f \circ \eta$ must be the identity map on I. So the monomorphism f has a section and is therefore an isomorphism.

We will be concerned with the case $\mathscr{C} = \textbf{\textit{Set}}$. Initial algebras do not always exist, not even in this case. For example, the functor

$$\begin{array}{rcl} \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & 2^X \end{array}$$

cannot not have an initial algebra, because in standard set theory it is not possible to have a bijection between a set and the set of its subsets. Note however that this functor is not polynomial. The main result of this chapter will be that for polynomial functors $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$, initial algebras always exist.

First let us have a look at some examples.

4.1.5 Example. Given a bijection $p : E \xrightarrow{\sim} B$, the corresponding polynomial functor is $P(X) = B \times X$. Hence a P-algebra is a set X equipped with an 'action' $B \times X \to X$. A P-algebra homomorphism is a map compatible with the actions

$$\begin{tikzcd} B \times X \arrow[r, "B\times f"] \arrow[d] & B \times Y \arrow[d] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

It is easy to see that $\emptyset$ is the initial P-algebra in this case.

4.1.6 Example. Generalising the previous example a little bit: for any surjection $E \to B$, the initial algebra for the corresponding polynomial functor is $\emptyset$. Indeed, $P(\emptyset) = \sum_{b \in B} \emptyset^{E_b}$, but all the summands are zero, because there are no empty fibres. Hence $\emptyset$ is a fixpoint, and it is obviously the least such. (In other words, if there are no nullary operations, then $\emptyset$ is the initial algebra.)

4.1.7 Example. Consider now the inclusion $p : \emptyset \to B$. The corresponding polynomial functor is the constant functor $P(X) = B$. A P-algebra is just a set equipped with a map from B, so the category of P-algebras is just the coslice category $B\backslash\textbf{\textit{Set}}$ of objects under B. The initial algebra is B equipped with the identity map.

Functoriality of least fixpoints

If $u : P \Rightarrow Q$ is natural transformation, there is induced a functor

$$\begin{aligned} u^* : Q\textbf{\textit{-alg}} &\longrightarrow P\textbf{\textit{-alg}} \\ [QX \to X] &\longmapsto [PX \overset{u_X}{\to} QX \to X] \end{aligned}$$

Note that this functor is the identity on the underlying sets.

In particular, if W_Q denotes a least fixpoint for Q, then u^*W_Q is a P-algebra. If W_P is a least fixpoint for P, i.e. an initial P-algebra, then we have a unique $P-$algebra map $W_P \to u^*W_Q$. Since u^* is the identity on the underlying sets, we may say that the assignment

$$\begin{aligned} \mathrm{End}(\mathscr{C}) &\longrightarrow \mathscr{C} \\ P &\longmapsto W_P \end{aligned}$$

is functorial.

4.2 Natural numbers, free monoids

4.2.1 Key example: construction of the natural numbers. Let $p : E \to B$ denote the inclusion of a one-element set into a two-element set—this is the simplest set map not of the two kinds treated in the previous examples. Then the corresponding polynomial functor is the free pointed-set monad,

$$P(X) = \{*\} + X.$$

The initial algebra for this polynomial functor is $\mathbb{N}$, *the set of natural numbers.* This is really the key example to understanding what the notion of initial algebras is about and what it has to do with induction, so we'll go through all the details, from various viewpoints.

4.2.2 Peano-Lawvere axiom for the natural numbers (called Dedekind-Peano axiom by Lawvere, see for example [72]). We continue with the polynomial functor $P(X) = \{*\} + X$. A P-algebra is a set X equipped with a set map $\{*\} + X \to X$, in other words, a set X with a distinguished element $x_0 \in X$ and an endomorphism $s : X \to X$. So P-algebras are diagrams $\{*\} \to X \to X$.

By the definition, an initial P-algebra is a P-algebra $\{*\} \to N \overset{\sigma}{\to} N$ such that for any other P-algebra (X, x_0, f) there is a unique $\varphi : N \to X$ making this diagram commute:

$$\begin{tikzcd}
& N \arrow[r, "\sigma"] \arrow[dd, "\varphi"'] & N \arrow[dd, "\varphi"] \\
* \arrow[ur] \arrow[dr] & & \\
& X \arrow[r, "f"'] & X
\end{tikzcd} \tag{4.1}$$

Peano-Lawvere

This universal property is precisely the *Peano-Lawvere axiom* for a natural number object in a topos. Now we do not want to assume any topos theory—and we like to wallow in details!—so let us explain this in terms of the usual Peano axioms for the natural numbers:

4.2.3 Peano's axioms for the natural numbers. The set of natural numbers can be characterised as a set N with a distinguished element $0 \in N$ and a successor function $s : N \to N$ satisfying

(i) 0 is not a successor

(ii) every element $x \neq 0$ is a successor

(iii) the successor function is injective.

(iv) If a subset $U \subset N$ contains 0 and is stable under the successor function, then $U = N$.

Note that (i)+(ii)+(iii) amount to saying that the map

$$\{*\} + N \xrightarrow{\langle 0,s \rangle} N$$

is a bijection. The last axiom is called the *induction axiom*.

(These axioms can be expressed in a more formal way, and in particular you can avoid using 'not-equal-to', but in our context the above formulation should suffice. . .)

4.2.4 Historical remark. These axioms are usually called the Peano axioms. They were introduced by Giuseppe Peano [88] in 1889, but in fact they were discovered by Richard Dedekind [31] in 1888 who stated them as a theorem. Peano explicitly acknowledges that he got the idea from Dedekind. The Peano-Lawvere axiom is the categorical reformulation of the axioms, as we shall now see.

4.2.5 The Peano axioms are equivalent to the Peano-Lawvere axiom (in *Set*). Suppose we are given $(N, 0, \sigma)$ satisfying Peano's axioms. We need to define a map $\varphi : N \to X$ making the diagram (4.1) commute. For the triangle to commute we need to set $\varphi(0) = x_0$, and for the square we need

$$\varphi(\sigma(n)) = f(\varphi(n)).$$

(in other words, $\varphi(n+1) = f(\varphi(n))$). Now Peano's axioms allow us to use these two conditions as *definition*—it is definition by induction! Indeed, by the first three axioms there is no contradiction in this definition, because every element in N is either 0 or the successor of a unique element, so we are not defining the value on a given element twice. Now the induction axiom ensures that this method exhausts N.

Conversely, given $(N, 0, \sigma)$ with the universal property. That is, it is initial among all diagrams $* \to X \to X$. Well, then Lambek's lemma tells us that $\{*\} + N \to N$ is an isomorphism, so this is already Peano's first three axioms! The remark about the terminology *least fixpoint* now expresses the last Peano axiom.

4.2.6 Finding an initial algebra for the above functor. Given any fixpoint, i.e. a set S with an isomorphism $f : \{*\} + S \xrightarrow{\sim} S$. (I.e., a set satisfying the first three axioms of Peano.) Then construct a set which satisfies all four axioms as follows. Define U to be the intersection of all subsets $Y \subset S$ satisfying $f(*) \in Y$ and $y \in Y \Rightarrow f(y) \in Y$. Then $(U, *, f)$ satisfies all Peano's axioms.

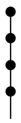
Perhaps we can describe U as the colimit (non-disjoint union)

$$0 \cup \{f(0)\} \cup \{f(f(0))\} \cup \ldots$$

The existence of a set S with $S \simeq 1 + S$ is called the *infinity axiom*. The argument shows that the infinity axiom is equivalent to the existence of $\mathbb{N}$.

4.2.7 Specific graphical version. CLEAN UP THIS GRAPHICAL EXAMPLE. To be specific, take $B = \{\texttt{stop}, \texttt{continue}\}$ and map the unique element of E to `continue`. MUCH BETTER TO STICK TO THE GRAPHICAL INTERPRETATION THAT WE KNOW AND WHICH WORKS SO WELL: THE ELEMENT `stop` IS THE NULLARY DOT, WHEREAS THE ELEMENT `continue` IS THE UNARY ONE-DOT OPERATION. THIS GIVES IMMEDIATELY THE INTERPRETATION WE WANT:

Now picture the empty set as a line. So $f(\emptyset)$ is either `stop` (the leafless one-dot tree) or `continue` (unary one-dot tree) followed by `stop`. The natural numbers are the union of all these trees, obtained by grafting on top. In each case there are two possibilities: `stop` or `continue`. But eventually we stop.



These pictures are really pictures of a certain level of repetition of this endofunctor—here there seems to be four dots, so this element comes in degree 4. The natural numbers is the colimit (the union) of all these possibilities. . .

4.2.8 Remark. Given this characterisation of $\mathbb{N}$, one can proceed to show that $\mathbb{N}$ is a monoid, in fact the free monoid on one generator. Now as such, there is an obvious generalisation to look for, namely the free monoid generated by any set S.

4.2.9 Constructing the free monoid on a set as an initial algebra. CLEAN UP HERE!! In order to generate $\mathbb{N}$, the free monoid on one generator 1, as least fixpoint, we took the polynomial functor represented by the set map $1 \to \{*\} + 1$. To get the free monoid on S, the set map should be

$$S \hookrightarrow \{*\} + S$$

Indeed, then there is one singleton fibre for each element in S, and in addition to that, one empty fibre. So the polynomial functor is

$$F_S(X) = 1 + S \times X.$$

4.2.10 Proposition. *The set $M(S)$ (the free monoid on S) is a least fixpoint for the endofunctor $F_S(X) = 1 + S \times X$.*

Proof. Recall first the definition of M: we have $M(S) = S^* = 1 + S + S^2 + S^3 + \cdots = \sum_{n \in \mathbb{N}} S^n$.

An F_S-set consists of an 'action' $S \times X \to X$ and a distinguished element $x_0 \in X$, and an F_S-map is one compatible with these structures. Now first check that the free monoid S^* is indeed an F_S-object: the action is

$$\begin{array}{rcl} S \times S^* & \longrightarrow & S^* \\ (s, x_1 \cdots x_k) & \longmapsto & s x_1 \cdots x_k \end{array}$$

and the distinguished element is the empty word $1 \in S^*$.

To check that S^* is an initial F_S-algebra is equivalent to showing that the free monoid on S is characterised by some Peano axiom like this: there is a bijection

$$1 + (S \times S^*) \simeq S^*.$$

Now suppose $K \subset S^*$ is a subalgebra. Let w be a word which is not in S^* and assume it is of minimal length. (It is not the empty word, because K contains the empty word just by being and F_S-algebra.) Now remove the first letter s, and let w' be the remainder of the word. Now this word is in the image of the successor map, namely the image of (s, w'), hence would have to be in K, hence $K = S^*$. So S^* is a least fixpoint.

□

4.2.11 Example. Generalising the natural numbers example a little bit, consider a pointed set $B = \{*\} + S$ (for some set S), with the natural augmentation map

$$\begin{array}{rcl} \mathbf{1} & \longrightarrow & B \\ * & \longmapsto & * \end{array}$$

The polynomial functor is

$$P(X) = \sum_{b \in B} X^{E_b} = X^1 + \sum_{s \in S} X^0 = S + X.$$

So P-algebras are diagrams $S \to X \to X$. Using Lambek iteration we see that the initial algebra is

$$\sum_{n\geq 0} S = \mathbb{N} \times S$$

Clearly there is an isomorphism $S + S \times \mathbb{N} \simeq S \times \mathbb{N}$—by distributivity $S + S \times \mathbb{N} \simeq S \times (1 + \mathbb{N})$ and we conclude by our favourite isomorphism $1 + \mathbb{N} \simeq \mathbb{N}$. The diagram is

$$\begin{tikzcd}
S \arrow[r] & S \times \mathbb{N} \arrow[r] & S \times \mathbb{N} \\
s \arrow[r, mapsto] & (s, 0) \arrow[r, mapsto] & (s, n+1)
\end{tikzcd}$$

End of the treatment of the natural numbers. Back to the general situation.

4.3 Tree structures as least fixpoints

initial

The trees we are going to get in this subsection are quite different from those used elsewhere in the book. They are trees without boundary. Usually our trees have input leaves and one output edge, which together form the boundary. In this section they have neither. We will call them *dead trees*, since they have no leaves, or *static trees* since they represent only nullary operations (i.e. constants).

4.3.1 Method for constructing the initial algebra. Given an endofunctor $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$, suppose it preserves monomorphisms and that it preserves sequential colimits. Note that all polynomial functors preserve monos (as a consequence of 1.7.1), and if it is defined by a finite map then it also preserves sequential colimits (by 1.7.3).

Start with the map $i : \emptyset \hookrightarrow P(\emptyset)$. Now take P on that and continue like this:

$$P^0(\emptyset) \overset{i}{\hookrightarrow} P^1(\emptyset) \overset{P(i)}{\hookrightarrow} P^2(\emptyset) \hookrightarrow \ldots$$

Since P preserves monos, all these maps are monos, so the colimit is just the union of all these sets. Now this union is the initial algebra: indeed, there is an isomorphism

$$P(\cup_{n\geq 0} P^n(\emptyset)) \simeq \cup_{n\geq 0} P(P^n(\emptyset)) = \cup_{n\geq 0} P^n(\emptyset)$$

This is Peano's first axiom. For the second we need to argue that no previous set in the sequence can work. Suppose we have a monomorphism $X \hookrightarrow \cup P^n(\emptyset)$ with $P(X) \simeq X$. Then we can find a section: start with $\emptyset \to X$, then we get $P(\emptyset) \to P(X) \simeq X$, and in the limit we get $\cup P^n(\emptyset) \to \cup P^n(X) \simeq X$. Check that this is a section. . .

4.3.2 Preview: general transfinite induction. The general case (the statement that every polynomial functor has an initial algebra) goes in the same way, except that there is no guarantee that the first colimit is a fixpoint! But then continue to apply the functor again and again and take the colimit of all that. This may once again not be the fixpoint, but then continue. Eventually this will stabilise. Doing this properly requires some nontrivial set theory, which we briefly review in the end of this section.

4.3.3 Planar binary trees. Consider the map $E \to B$:

$$\begin{array}{rcl} \{\texttt{left},\texttt{right}\} & \longrightarrow & \{\texttt{stop},\texttt{continue}\} \\ \texttt{left} & \longmapsto & \texttt{continue} \\ \texttt{right} & \longmapsto & \texttt{continue} \end{array}$$

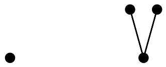
There is one empty fibre and one fibre of cardinality 2. In other words, one nullary operation and one binary. Hence the polynomial functor is $P(X) = 1 + X^2$.

We picture the base set B as consisting of these two operations:

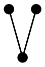
We claim the initial algebra (i.e. the least fixpoint) for this polynomial functor is the set of planar binary trees.

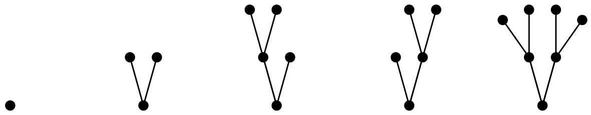
To compute it, start with $P(0)$: by the graphical approach to polynomial functors, this is the set of all ways of decorating the two bouquets by elements in the empty set. Since there is precisely one nullary operation, we have $P(0) = \{\bullet\} = 1$.

Next, $P(1)$ is just the base set B, by the standard interpretation 1.1.8, but in this case the singleton 1 is the specific set $\{\bullet\}$ consisting of one nullary operation, so we are better off keeping within the strictly graphical interpretation: hence $P(\{\bullet\})$ is the set of all ways of decorating either of the two bouquets in B by elements in $\{\bullet\}$, so the picture of the two possibilities becomes



We should also describe the map $P(0) \hookrightarrow P(1)$: it is of course just the inclusion of $\bullet$ into $P(1)$ mapping the nullary bouquet into the nullary bouquet. Indeed, since the empty set 0 is a subset of $P(0)$, decorating in 0 is a subset of decorating in $P(0)$.

Next, $P(B)$ consists of the nullary operation together with all ways of decorating the two-leaf bouquet with • or  This gives 5 possibilities altogether:



Again the inclusion of B^1 into B^2 is precisely the one that can be seen in the drawings: i.e. the two figures in B^1 are included in B^2 as the same two figures. (Which is just a sign that the drawing conventions are good. . .)

In the next step we get the $1 + 5^2 = 26$ binary tree of height at most 4, and next the $1 + 26^2 = 677$ binary trees of height at most 5. Clearly the conclusion is that the union of all that is the set of all binary planar dead trees (i.e. without free input leaves).

So the least fixpoint for this polynomial functor is the set of binary planar dead trees. The planarity comes about because the fibre was $\{$`left,right`$\}$. This was sort of cheating or smart, in order to take advantage of the orientation of the paper. More abstractly, the fibre could be any two-element set, for example $\{$`red,blue`$\}$, and in that case we would have to specify for each node which input edge is red and which is blue. It is clear, though, that we could get an isomorphic set of trees. Hence by abuse of concepts we say that we get the planar binary trees. But in fact it can often be very helpful not to be bound to using ordered sets as labels. We shall see later that more formally we are talking about *P-trees*.

The equation satisfied by planar binary trees is

$$X = 1 + X^2$$

which expresses that a planar binary tree is either the trivial tree, or otherwise a pair of planar binary trees.

Notice how the possible node shapes are precisely listed in the cograph of the map $E \to B$. (See Lawvere–Rosebrugh [72] for the notion of cograph.)

4.3.4 Planar trees. Start with our favourite map $\mathbb{N}' \to \mathbb{N}$, representing the free monoid endofunctor M,

$$X \mapsto \sum_{n \geq 0} X^n$$

So there is one operation of each arity n, and each is interpreted as a bouquet with n inputs. Combining these freely, iterating the operations, we construct all planar trees (without input leaves). It is easy to see that the set T of all such planar trees is a fixpoint. This is just to follow the same construction as we just applied to planar binary trees.

The last observation reveals something I don't understand: we have an isomorphism

$$T \simeq \sum_{n \geq 0} T^n$$

expressing the characterisation that a tree is a sequence of trees, possibly the empty sequence. Now, in functional terms: we have the equation $T = \frac{1}{1-T}$ which we REWRITE (CAREFUL HERE!) as

$$T = 1 + T^2.$$

This is weird! it seems to be the same equation as for binary trees! Well it is true that any planar binary tree is a planar tree, and hence it might be another fixpoint for the same functor... one of these is the least one... when rewriting, perhaps we did some assumption... one equation is perhaps a factor in the other...)

THERE IS SOME PAPER BY D. KNUTH ABOUT TREES=BNINARY-TREES??? WHERE DID I SEE THIS MENTIONED?

4.3.5 Variations. A very interesting variation of the previous examples it to consider, for any polynomial functor P the functor $1 + P$. The extra constant should be considered the polynomial functor represented by $0 \to 1$, so the unique element in 1 is a nullary operation (as always, we consider fixed sets as sets of nullary operations). Repeating now the colimit construction for $1 + P$ we get the following least fixpoint: its elements are the trees built out of the operations in P and the extra nullary operation from 1, and no free leaves are allowed. If we think of the new nullary operation as a red dot, then we get the same trees as before but with some of the stopdots coloured red.

More generally, if we fix a set A, as always thought of as a set of nullary operations, then the least fixpoint for $A + P$ is the set of dead trees with stopdots either in A or in the set of nullary operations of P.

BE EXPLICIT ABOUT THE TWO EXAMPLES, BINARY TREES AND GENERAL PLANAR TREES.

Continuing with the example of planar binary tree, and the example M giving all planar trees, we can easily figure out what the least fixpoints are for $A + B$ or $A + M$. Let's just work with $A + M$.

Fix(A+M)

4.3.6 Least fixpoint for $1 + M$**: live trees.** Let us compute the least fixpoint for $X \mapsto A + M(X)$. This is a key example. The constant polynomial endofunctor with value A is represented by the map $\emptyset \to A$, so the set A is a set of nullary operations. Hence the set of operations of $A + M$ is the set consisting of the bouquets (one for each natural number) and then an extra collection of nullary operations, one for each element in A. Repeating the colimit construction of the least fixpoint, the first step gives us that $(A + M)(\emptyset)$ is the set of coloured dots: there is one neutral colour (corresponding to the nullary operation in M) and one dot coloured a for each element $a \in A$. You immediately see that the least fixpoint is the set of dead trees, in which each stopdot (nullary node) has a colour.

But in fact, the example $A = 1$ is the most important. EXPLAIN WHY WE WANT TO USE THE SEPCIAL-PURPOSE SINGLETON SET $1 = \{\texttt{blank}\}$.

In a moment we are going to let A vary, to construct a left adjoint to the forgetful functor P-***alg*** $\to$ ***Set***.

4.4 Induction, well-founded trees

The free-monoid examples (and $\mathbb{N}$ in particular) illustrate the importance of least fixpoints. We see that the notion of fixpoint for an algebra is a generalisation of the first three axioms of Peano. Being in addition the *least* fixpoint generalises the induction axiom. We shall formalise the analogy further.

4.4.1 Definition of wellfounded trees. In the category of sets, they are just trees with the property that there are no infinite paths. This is a stronger condition than just saying that every node is at finite distance from the root, in which case the tree as a whole could have infinite height.

The signature of a tree (also called branching type) is the set of possible kinds of nodes the tree can have, i.e. what sort of structure is considered on the input edges of the nodes. For example the kind could be just 'binary', or there could be allowed arbitrary finite number of incoming edges at each node, or various infinities could be allowed. In other words, a signature is a certain set of bouquets. And in other words, a signature is just a polynomial functor $E \to B$, where B is the set of bouquets, and the fibre over each $b \in B$ is the set of incoming edges to that node.

Hence a wellfounded tree of signature B is a tree that can be obtained by grafting together bouquets from B in a finite number of steps (but without bounding this finiteness).

In conclusion, the set of wellfounded trees of signature $E \to B$ is precisely a least fixpoint for the polynomial functor $E \to B$. Or equivalently, as we shall see in the next section: wellfounded trees of signature $E \to B$ are the operations for the free monad on $E \to B$ (CHECK THAT THIS IS CORRECT, OTHERWISE SOMETHING VERY SIMILAR IS TRUE.)

4.4.2 Induction and wellfoundedness. One way to think of a P-algebra X is as a set X equipped with a family of operations

$$\big(\mu_b : X^{E_b} \to X \mid b \in B\big)$$

so in the natural numbers example there was one nullary and one unary operation.

If the initial algebra for a polynomial functor $p : E \to B$ exists it is denoted $W(p)$, the set of wellfounded trees of signature p. Of course the set $W(p)$ depends crucially on p, but in the following couple of paragraphs we hold p fixed and suppress p from the notation, writing $W := W(p)$ for simplicity. Since W is a P-algebra, for each $b \in B$ there is an operation denoted

$$\sup_b : W^{E_b} \to W$$

Initiality of W means that for any P-algebra X, there is a unique P-algebra map $W \to X$. In the fibre-wise description this means that for any P-algebra $(\mu_b : X^{E_b} \to X \mid b \in B)$ there is a unique map $\varphi : W \to X$ such that the diagrams

$$\begin{tikzcd}
X^{E_b} \arrow[r, "\mu_b"] & X \\
W^{E_b} \arrow[u, "\varphi^{E_b}"] \arrow[r, "\sup_b"'] & W \arrow[u, "\varphi"']
\end{tikzcd}$$

commutes for every $b \in B$. This is to say that for every $t : E_b \to W$ we have

$$\varphi(\sup_b(t)) = \mu_b(\varphi \circ t)$$

This expresses that φ is defined by induction: if we already know the values of φ on the image of all $t : E_b \to W$ then the equation defines also φ on $\sup_b(t)$.

In intuitive terms, we have all the trees pointed to by the map $E_b \to W$. Their sup is obtained by gluing all those trees onto the leaves represented by the elements in E_b. (Every nontrivial tree has a root node and a collection of all the ideal subtrees given by the input edges of that root node.)

By Lambek's lemma, $P(W) \to W$ is an isomorphism. This means that each $w \in W$ is the image of a unique $W^{E_b} \to W$ for unique $b \in B$ and unique $t : E_b \to W$. In other words, each $w \in W$ is of the form $\sup_b(t)$ for unique $b \in B$ and $t : E_b \to W$.

This is the generalisation of Peano's first three axioms: for $\mathbb{N}$ the statement is that each natural numbers is of the form $f(b)$ for a unique $b \in \{stop, continue\}$ and $t : E_b \to B$—this is just to say that every element is either 0 or a successor.

As we noted, initiality of W implies that every subalgebra of W must be W itself. This can be stated like this: if $R \subset W$ is such that for every $b \in B$ and every $t : E_b \to W$, whenever the image of t is inside R then $\sup_b(t)$ is also in R. Then $R = W$.

4.5 Transfinite induction

THE TRANSFINITE INDUCTION PROOF THAT EVERY POLYNOMIAL ENDOFUNCTOR HAS AN INITIAL ALGEBRA—look up details in Borceux.

4.5.1 Ordinals. The ordinals can be characterised as the smallest set with successor function and supremum. This means that every element has a successor, and that for every small subset there is a supremum—a smallest element greater than everybody in the subset. What about the order? Is it assumed or is it a consequence of the two other structures?

In practice, start with the empty subset of ***Ord***. It must have a supremum which we call 0. Now take the successive successors of 0, to get all finite ordinals. This is a small set (we will not go into the subtleties of the technical meaning of the word 'small', but let us just promise that countable is small) so it has a supremum, which we call ω. Repeat like this, alternating between creating new elements using successor and supremum. (Note that this will give us two sorts of ordinals: successors and limits).

$$0, 1, 2, \ldots \omega, \omega + 1, \omega + 2, \ldots \omega{\cdot}2, \omega{\cdot}2 + 1 \ldots \omega^2, \ldots \omega^3, \ldots, \omega^\omega, \ldots \omega^{\omega^\omega}$$

Remark: The limit of all those $\omega^{\cdot^{\cdot^{\omega}}}$ is a famous ordinal called ε_0. It is the first ordinal number that cannot be constructed from smaller ones by finite additions, multiplications, and exponentiations. It is also the first ordinal that satisfies the equation $x = \omega^x$.

However, the list of ordinals just goes on forever...

Every ordinal can be written in a unique way on Cantor normal form, as a finite sum of smaller and decreasing ordinals:

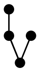
$$\alpha = \alpha_1 + \alpha_2 + \cdots + \alpha_n \qquad \text{with } \alpha_1 \geq \cdots \geq \alpha_n.$$

It is a little bit analogous to the obvious fact that every natural number can be written in base 10, i.e. as a finite sum of ... 10000, 1000, 100, 10, 1.

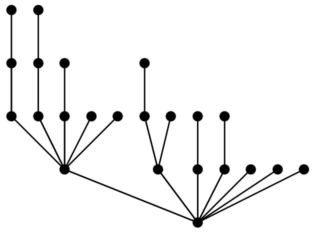
For the ordinals smaller than ε_0, the necessary ingredients are just 0, ω, sum, and exponentiation. We use the equation $\omega^0 = 1$. For example here is a certain ordinal on Cantor normal form

$$\omega^{\omega^\omega + \omega^\omega + \omega + 1 + 1} + \omega^{\omega + 1} + \omega + \omega + 1 + 1 + 1.$$

Recalling that $1 = \omega^0$ we see that it is just a finite sum of omegas, raised to different exponents, which in turn are smaller ordinals and therefore are finite sums of omegas raised to different exponents, and that every exponentiation terminates with 0 as the most deeply nested exponent.

In conclusion, every ordinal smaller than ε_0 is represented by a finite rooted tree: a dot represents a sum, and its children are the summands. In particular a dot without children is the empty sum 0. Each edge represents ω, and the subtree sitting on top of the edge represents its exponent. So for example the tree • represents the empty sum 0. The tree represents the one-term sum $\omega^0 = 1$, the tree  represents $\omega^{\omega^0} + \omega^0 = \omega + 1$.

The example of normal form given above is represented by the tree



These are non-planar trees, but when converting a tree into a normal-form ordinal we have to order the children of each node according to the order of the subtrees...

The ordinal ε_0 was used by Gentzen [40] to prove the consistency of arithmetics. (We know by Gödel's theorem that one cannot prove the consistency of arithmetics within arithmetics. But by going out to a large number system Gentzen was able to prove the consistency. His proof uses induction indexed by ε_0... somehow indexing all possible expressions of arithmetics??)

Now being an ordered set, ***Ord*** is just a special case of a category, supremum is just a special case of colimit. Now let $\mathscr{C}$ be a cocomplete category equipped with an endofunctor $F : \mathscr{C} \to \mathscr{C}$. Then there is a canonical functor ***Ord*** $\to$ ***Cat*** which preserves colimits and sends the successor function to F.

4.5.2 Finer theory of colimit preservation. We already observed that if F preserves sequential colimits, then after iterating ω times we stabilise. More generally let α be an ordinal, then an α-chain in $\mathscr{C}$ is a chain indexed by chains shorter than α. Then there is some sort of notion of preserving α-colimits, and that will ensure that the iteration stabilises after α.

Now given a polynomial functor, given by a set map $p : E \to B$. If all fibres are finite, then P will preserve ω-chains.

If the biggest fibre E_b is of some cardinality w, then there will be a least ordinal σ dominating w, and then P will preserve σ-sequential colimits.

4.5.3 Theorem. *Every polynomial endofunctor has a least fixpoint.*

XXXXAnd in fact we do, although we have not yet proved this. If we assume P preserves sequential colimits (e.g. P is represented by a finite map) then the least fixpoint is the object

$$\bigcup_{n \in \mathbb{N}} P^n(\emptyset).$$

This is then $U(F_P(\emptyset))$. So if just we make explicit what its algebra structure is, then we have discovered the value of F_P on $\emptyset$. The algebra structure is the natural isomorphism

$$P(\bigcup_{n \in \mathbb{N}} P^n(\emptyset)) \longrightarrow \bigcup_{n \in \mathbb{N}} P^n(\emptyset)$$

4.5.4 Example. Recall from 4.3.6 that if P is the free-monoid monad ***Set*** $\to$ ***Set*** then the least fixpoint is the set of dead trees. In other words, $F(\emptyset) =$ trees. And also that the least fixpoint for $A + M$ is the set of dead trees some of whose dot-leaves are decorated with elements in A.

In general, this same descriptions holds, and the graphical interpretation is clear: the least fixpoint is the set of all dead P-trees, or closed P-trees: these are all the trees that can be built from the bouquets representing the set of operations B, with the condition that there are no open leaves left. The condition that everything is closed comes from the colimit construction: already in the first step we only get contribution from nullary operations, since these are the only ones that can be decorated in the empty set. So already here we only get closed stuff, and from here on we only ever use these as decorations.

We now aim at computing $F_P(A)$ for any object A. In a minute we shall see that the underlying set of $F_P(A)$ is a least fixpoint for the endofunctor $X \mapsto A + PX$, so we can compute $F_P(A)$ as the union

$$\bigcup_{n \in \mathbb{N}} (A + P)^n(\emptyset).$$

Now we can repeat the graphical description: the constant A is given by the polynomial functor represented by the map $\emptyset \to A$, so all the elements in A must be interpreted as nullary operations. So in the first step in the colimit construction, computing $P(A)$ we get the union of A with the set of nullary operations of P. And in the end we get the set of all trees formed by the operations in P together with the operations in A (all nullary), with the condition that no leaves are left open.

4.6 Free-forgetful

From now on we let $P : \boldsymbol{Set} \to \boldsymbol{Set}$ denote a polynomial functor. Most of the constructions work for more general endofunctors, but then some conditions should be put here and there... We also restrict attention to the case $\mathscr{C} = \boldsymbol{Set}$. WE SHOULD RESTRICT ATTENTION TO POLYNOMIAL ENDOFUNCTORS

The idea is that the least-fixpoint construction assembles into a functor, and this functorial construction provides us with better and more dynamic trees. Namely, with the help of the least-fixpoint construction we get a left adjoint F to the forgetful functor $U : P\text{-}\boldsymbol{alg} \to \mathscr{C}$.

This adjoint pair generates a monad $T := U \circ F : \boldsymbol{Set} \to \boldsymbol{Set}$ the *free monad* on P. It has a very explicit description in terms of decorating trees, and we see that it is in fact a polynomial monad! The operations for this polynomial monad are the good dynamic trees.

alg-adj

4.6.1 The free P-algebra functor. The forgetful functor $U : P\text{-}\boldsymbol{alg} \to \mathscr{C}$ has a left adjoint F_P, the free P-algebra functor. This is very general—perhaps there is a concrete description of it. We will rather assume it exists, and then slowly discover how it goes.

Suppose $\mathscr{C}$ has an initial object $\emptyset$. Since F_P is a left adjoint it preserves initial objects, so $F_P(\emptyset)$ is an initial algebra, and $U(F_P(\emptyset))$ is a least fixpoint. So if we know the left adjoint, we can use it to compute least fixpoints.

Conversely, we will now assume we know how to construct the least fixpoint.

4.6.2 Constant endofunctors. Consider now a constant endofunctor $X \mapsto A$. Then an A-algebra is just an arrow $A \to X$, and an A-algebra map is just a triangle

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ X \arrow[rr] & & Y \end{tikzcd}$$

In other words, the category of A-algebras is naturally identified with the coslice category $A/\mathscr{C}$. In this case it is clear what the free A-algebra on an object X is: it is just $A + X$, assuming that $\mathscr{C}$ has sums. Indeed, let us check the bijection

$$\mathrm{Hom}_{\mathscr{C}}(X, U(R)) \simeq \mathrm{Hom}_{A\text{-}\boldsymbol{alg}}(A + X, R),$$

where $A \rightarrow R$ is an A-algebra and X is an object of $\mathscr{C}$: but to give an A-map $A + X \rightarrow R$ is the same as giving just a map $X \rightarrow R$.

4.6.3 Translations of P. We fix an endofunctor $P : \mathscr{C} \rightarrow \mathscr{C}$, and study its translations,

$$\begin{array}{rcl} P_A := A + P : \mathscr{C} & \longrightarrow & \mathscr{C} \\ X & \longmapsto & A + PX. \end{array}$$

There is a cartesian diagram of forgetful functors

$$\begin{tikzcd} (A{+}P)\text{-}\boldsymbol{alg} \arrow[r] \arrow[d] & P\text{-}\boldsymbol{alg} \arrow[d] \\ A\text{-}\boldsymbol{alg} \arrow[r] & \mathscr{C} \end{tikzcd}$$

A P_A-algebra is an object K with a map $A + PK \rightarrow K$. In other words, is a triple (K, a, s) where K is an object of $\mathscr{C}$, $s : P(K) \rightarrow K$ gives it P-algebra structure, and $a : A \rightarrow K$ gives it A-structure.

4.6.4 Proposition. *$U_P : P$-**alg** has a left adjoint if and only if for every set A, the polynomial endofunctor $A + P$ has a least fixpoint.*

Proof. The category $(A{+}P)$-**alg** is naturally identified with the comma category ${}_A\backslash P$-**alg** which we now proceed to describe. The result follows from general facts: □

4.6.5 Reminder on comma categories and adjunctions. (See also Mac Lane [76], Theorem 2 on page 83.) Let $G : \mathscr{D} \to \mathscr{C}$ be a functor, and let X be an object of $\mathscr{C}$. The *comma category* $X \downarrow G$ is the following. Its objects are pairs (D, δ) where D is an object of $\mathscr{D}$, and $\delta : X \to GD$ is an arrow in $\mathscr{C}$. An arrow from (D', δ') to (D, δ) is an arrow $\alpha : D' \to D$ in $\mathscr{D}$ such that

$$\begin{tikzcd}
 & X \arrow[dl, "\delta'"'] \arrow[dr, "\delta"] & \\
GD' \arrow[rr, "G\alpha"'] & & GD.
\end{tikzcd}$$

An object (I_X, η_X) in $X \downarrow G$ is initial when for each $\delta : X \to GD$ there is a unique $\alpha : I_X \to D$ such that

$$\begin{tikzcd}
 & X \arrow[dl, "\eta_X"'] \arrow[dr, "\delta"] & \\
GI_X \arrow[rr, "G\alpha"'] & & GD.
\end{tikzcd}$$

In other words, the map

$$\begin{array}{rcl} \mathscr{D}(I_X, D) & \longrightarrow & \mathscr{C}(X, GD) \\ \alpha & \longmapsto & G\alpha \circ \eta_X \end{array} \tag{4.2}$$

is a bijection.

Suppose now that for every object $X \in \mathscr{C}$, the comma category $X \downarrow G$ has an initial object (I_X, η_X). Then we can construct a functor $F : \mathscr{C} \to \mathscr{D}$ which is left adjoint to G. On objects, we set $F(X) := I_X$. On arrows, given $\phi : X' \to X$ in $\mathscr{C}$, consider the diagram

$$\begin{tikzcd}
X' \arrow[r, "\phi"] \arrow[d, "\eta_{X'}"'] & X \arrow[d, "\eta_X"] \\
GI_{X'} \arrow[r, dashed] & GI_X.
\end{tikzcd}$$

Initiality of $(I_{X'}, \eta_{X'})$ tells us there is a unique arrow $\psi : I_{X'} \to I_X$ such that $G(\psi)$ fits in as the dashed arrow. We define $F(\phi) := \psi$. It is easy to see that this makes F into a functor. The diagram now reads

$$\begin{tikzcd}
X' \arrow[r, "\phi"] \arrow[d, "\eta_{X'}"'] & X \arrow[d, "\eta_X"] \\
GFX' \arrow[r, "F\phi"'] & GFX,
\end{tikzcd}$$

showing that the arrows η_X assemble into a natural transformation $\eta : \mathsf{Id}_{\mathscr{C}} \Rightarrow GF$. The bijection (4.2) now says that F is left adjoint to G.

Conversely, given a left adjoint $F \dashv G$ with unit η, the arguments above show that the pair (FX, η_X) is an initial object of the comma category $X \downarrow G$.

In conclusion we have proved:

4.6.6 Lemma. *A functor $G : \mathscr{D} \to \mathscr{C}$ has a left adjoint if and only if for each object $X \in \mathscr{C}$, the comma category $X \downarrow G$ has an initial object.* □

To finish the proof of the Proposition, take $\mathscr{D} = P$***-alg***. It is clear from the general description of the objects and arrows in $A \backslash \mathscr{E}$ that it is naturally identified with $(A+P)$***-alg***, hence the proposition follows.

A few ad hoc arguments: suppose we know (for each X) that W_X is the initial $(X+P)$-algebra. By the inclusion $P \subset X + P$ it is also a P-algebra. Let us describe explicitly the bijection

$$P\textbf{\textit{-alg}}(W_X, A) \simeq \textbf{\textit{Set}}(X, UA).$$

Given $X \to UA$, that makes A into an X-algebra. Since it is already a P-algebra, it therefore becomes an $X+P$-algebra, and since W_X is the initial such, we have a map $W_X \to A$. Conversely, given a P-algebra map $W_X \to A$, the fact that W_X is an $X+P$-algebra gives us further a map $X + P(W_X) \to W_X \to A$, and hence in particular a map $X \to A$.

4.6.7 When P is a monad. We already observed that when P is a monad, then $P(X)$ is naturally a P-algebra. In that case there is a natural map of P-algebras $F(X) \to A(X)$ (natural in X). Indeed, by adjunction

$$\mathrm{Hom}_{P\textbf{-alg}}(FX, PX) \simeq \mathrm{Hom}_{\mathscr{C}}(X, PX),$$

and in here we have $\eta_X : X \to PX$.

(You might also try to find this map directly by finding a map $\cup (A + P)^n(\emptyset) \to P(A)$. In an case you need the unit.

See Theorem 2 on page 311 of Barr & Wells [13].

4.6.8 The functor T_P. We now know the value of T_P on any set A: it is the least fixpoint for the functor $A + P$, and we have described it explicitly as the set of trees made from the operations in P, together with special stopdots decorated in A.

Now we vary the set A. We should also indicate what T_P does on arrows: given a set map from $\varphi : A \to B$, the image arrow is simply the map that sends a trees with A-stopdots to the same tree with B-stopdots, replacing a stopdot $a \in A$ by $\varphi(a)$.

Since we are now varying the decorating set A, we might perhaps rather think of those variable decorations as empty slots where any set A can be thrown in. So what is the generic set of trees? Well, that's just $T_P(1)$, where like in 1.5.3 we think of the label 1 as `blank`. Since we leave the input open, we represent it by a dotless line.

Slogan: 1 *means blank*

So we think of $T(1)$ as the set of trees made out of the operations in P and one extra operation, which is in fact nullary, but since we are going to use it for decorations, we prefer to think of it as an open leaf. So there is no longer any requirement that all leaves are closed. If the original polynomial functor is represented by $E \to B$, denote by $B^* := T(1)$, the set of these generic, dynamic trees. Let E^* denote the set of such trees with a marked input leaf, i.e. one of the 1-stopdots singled out.

The conclusion of this whole discussion is

4.6.9 Theorem. *T_P itself is a polynomial functor, represented by $E^* \to B^*$.*

Indeed, T_P applied to a set A is the set of all ways to decorate the generic trees $T_P(1)$ by elements in A. This is the description we found in the construction of T_P, and it is precisely the description of the polynomial functor $E^* \to B^*$.

This new polynomial functor T_P, the free monad on P: in general there is no map $T_P \to P$. These maps are in one-to-one correspondence with maps $B^* \to B$. To have such a map we need extra structure: in fact if P is a monad then we do precisely have such a map!

4.6.10 Example. If M : ***Set*** $\rightarrow$ ***Set*** is the free-monoid monad. The value of M on a set X is the set of planar bouquets with leaves decorated in X. An M-algebra is a set X equipped with a map $\sum_{n\in\mathbb{N}} X^n \rightarrow X$. The free M-algebra functor associates to a set X the set of planar trees with leaves decorated in X. Note that in $M(2)$ there are two different two-leaf bouquets with leaves decorated in 2: because E_b is a specified set, and we study maps $E_b \rightarrow X\ldots$ XXXXXXXXXXXX

????This is precisely the polynomial functor represented by $E^* \rightarrow B^*$.

4.6.11 Remark. Important remarks: let U : P***-alg*** $\rightarrow$ ***Set*** denote the forgetful functor from the category of Lambek algebras. The category of Lambek algebras is often denoted $(P$: ***Set***$)$. If this functor has a left adjoint F then it is monadic. This means that P***-alg*** is equivalent to the Eilenberg-Moore category $\textbf{\textit{Set}}^T$, where $T := U \circ F$. [13, Ch.9,Prop.1] [THAT SHOULD BE PROPOSITION 9.4.5]. Furthermore, in this case $T := U \circ F$. is the free monad on P [13, Ch.9,Thm.3]. [THAT SHOULD REFER TO THEOREM 9.4.4] [BOTH REFS, P.269]

Fixpoint construction via bar-cobar duality

Assume that $\mathscr{C}$ has filtered colimits. Consider any endofunctor $P : \mathscr{C} \rightarrow \mathscr{C}$.

Let (C, c) be a P-coalgebra. That is, an object C together with a morphism $c : C \rightarrow PC$. Then we can construct a fixpoint for P as follows. Consider the sequence

$$C \xrightarrow{c} PC \xrightarrow{Pc} P^2C \xrightarrow{P^2c} P^3C \longrightarrow \cdots$$

Put

$$I := I_C := \operatorname*{colim}_{n\rightarrow\infty} P^nC$$

This is naturally a P-coalgebra again. Indeed, whether or not P preserves filtered colimits, there is always a map

$$PI = P(\operatorname*{colim}_n P^nC) \longleftarrow \operatorname*{colim}_n P^{n+1}C \simeq I$$

Assume now that $P : \mathscr{C} \rightarrow \mathscr{C}$ preserves filtered colimits. Then the structure map $PI \leftarrow I$ is invertible, so I becomes a fixpoint, and hence also a P-algebra, and indeed a fixpoint, by taking the structure map

$$PI = P(\operatorname*{colim}_n P^n C) \simeq \operatorname*{colim}_n P(P^n C) \simeq \operatorname{colim} P^n C \simeq I.$$

It is easy to check that if $(C, c) \to (D, d)$ is a P-coalgebra homomorphism, then the induced map $I_C \to I_D$ is a P-algebra homomorphism.

Altogether, we have constructed a functor

$$P\textbf{-}\textbf{\textit{coalg}} \to P\textbf{-}\textbf{\textit{alg}}$$

which actually takes values in fixpoints.

Remark: how to get, not just a fixpoint, but actually an initial algebra? Well, just input the canonical coalgebra $\emptyset$ (assuming that the category $\mathscr{C}$ has an initial object).

Recall that to define a morphism out of a colimit is to pick a 'stage' in the colimit diagram, and then define a map from there. (Maybe a more careful description is needed in general, but for the sequential colimit this is OK). Two such maps $f : I_n \to X$ and $g : I_m \to X$ define the same map $I \to X$ when

$$\begin{tikzcd}
I_m \arrow[r] \arrow[drr, "g"'] & \cdots \arrow[r] & I_n \arrow[d, "f"] \\
& & X
\end{tikzcd}$$

commutes.

Now we shall see how to define P-algebra maps out of fixpoints of the kind coming from P-coalgebras.

First a definition that may look silly. A *twisted morphsm* from a P-coalgebra (C, c) to a P-algebra (A, a) is a morphism $f : C \to A$ such that this square (of maps in $\mathscr{C}$) commutes:

$$\begin{tikzcd}
C \arrow[r, "c"] \arrow[d, "f"'] & PC \arrow[d, "Pf"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

Formally, the set of twisted morphisms from C to A is defined as the equaliser

$$\operatorname{Tw}(C, A) \to \operatorname{Map}(C, A) \rightrightarrows \operatorname{Map}(C, A)$$

where the two maps $\operatorname{Map}(C, A) \to \operatorname{Map}(C, A)$ are the identity and $f \mapsto a \circ P(f) \circ c$.

If (C, c) is a P-coalgebra, then (PC, Pc) becomes a P-coalgebra.

4.6.12 Lemma. *The natural map*

$$\begin{array}{rcl}
\operatorname{Tw}(C, A) & \longrightarrow & \operatorname{Tw}(PC, A) \\
f & \longmapsto & a \circ Pf
\end{array}$$

is an equivalence (here just a bijection). The inverse is given by

$$g \circ c \longleftarrow\!\shortmid g$$

Proof. We should first make explicit how $a \circ Pf$ is a twisted morphism. Starting with the diagram that says that f is a twisted morphism:

$$\begin{tikzcd}
C \arrow[r, "c"] \arrow[d, "f"'] & PC \arrow[d, "Pf"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

apply P and paste with an obvious square like this:

$$\begin{tikzcd}
PC \arrow[r, "Pc"] \arrow[d, "Pf"'] & PPC \arrow[d, "PPf"] \\
PA \arrow[d, "a"'] & PPA \arrow[l, "Pa"] \arrow[d, "Pa"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

The right-hand composite is precise $P(a \circ Pf)$ as required.

In the other direction, given a diagram for g

$$\begin{tikzcd}
PC \arrow[r, "Pc"] \arrow[d, "g"'] & PPC \arrow[d, "Pg"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

just paste on top of it like this:

$$\begin{tikzcd}
C \arrow[r, "c"] \arrow[d, "c"'] & PC \arrow[d, "Pc"] \\
PC \arrow[r, "Pc"] \arrow[d, "g"'] & PPC \arrow[d, "Pg"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

To see that the two constructions are inverse: starting with the square for f, going right and then back left gives

$$\begin{tikzcd}
C \arrow[r, "c"] \arrow[d, "c"'] & PC \arrow[d, "Pc"] \\
PC \arrow[r, "Pc"] \arrow[d, "Pf"'] & PPC \arrow[d, "PPf"] \\
PA \arrow[d, "a"'] & PPA \arrow[l, "Pa"] \arrow[d, "Pa"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

but this is precisely the original square for f.

On the other hand, starting with the square for g, going left and then back right gives

$$\begin{tikzcd}
PC \arrow[r, "Pc"] \arrow[d, "Pc"'] & PPC \arrow[d, "PPc"] \\
PPC \arrow[r, "PPc"] \arrow[d, "Pg"'] & PPPC \arrow[d, "PPg"] \\
PA \arrow[d, "a"'] & PPA \arrow[l, "Pa"] \arrow[d, "Pa"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

but this is precisely the original square for g. □

4.6.13 Proposition. *To give a P-algebra homomorphism $(I_C, i) \to (A, a)$ is equivalent to giving a twisted morphism $(C, c) \to (A, a)$.*

4.6.14 Proposition. *To give a P-algebra homomorphism $f : (I, i) \to (A, a)$ is equivalent to giving for every $n \in \mathbb{N}$ a morphism*

$$f_n : I_n \to A$$

such that

$$\begin{tikzcd}
I_n \arrow[r, "jn"] \arrow[d, "f"'] & I_{n+1} \arrow[d, "Pf"] \arrow[dl, "f_{n+1}"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

Proof. We wish to calculate $\mathrm{Map}(i, a)$. The first step is to see that this is the equaliser

$$\mathrm{Map}(i, a) \longrightarrow \mathrm{Map}(I, A) \rightrightarrows \mathrm{Map}(PI, A)$$

so we are after the space of maps $f : I \to A$ such that

$$\begin{tikzcd}
I & PI \arrow[l, "i"'] \arrow[d, "Pf"] \\
A \arrow[from=1-1, "f"'] & PA \arrow[l, "a"]
\end{tikzcd}$$

Now since i is invertible, this is the same as the space of maps $f : I \to A$ such that

$$\begin{tikzcd}
I \arrow[r, "i^{-1}"] \arrow[d, "f"'] & PI \arrow[d, "Pf"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

i.e. the space of twisted morphisms from I to A. This we know is described as the equaliser

$$\mathrm{Tw}(I, A) \to \mathrm{Map}(I, A) \rightrightarrows \mathrm{Map}(I, A) \qquad (4.3) \quad \boxed{\texttt{TwIA}}$$

where the two maps $\mathrm{Map}(I, A) \to \mathrm{Map}(I, A)$ are the identity and $f \mapsto a \circ P(f) \circ i^{-1}$.

Now these mapping spaces are

$$\mathrm{Map}(I, A) = \mathrm{Map}(\operatorname*{colim}_n P^n C, A) = \lim_n \mathrm{Map}(P^n C, A)$$

Altogether we can write down a big commutative diagram

$$\begin{tikzcd}
{} & {} \\
\mathrm{Map}(P^n C, A) \arrow[u] \arrow[r, shift left, "\mathrm{id}"] \arrow[r, shift right, "a\circ P(\ )\circ P^n(c)"'] & \mathrm{Map}(P^n C, A) \arrow[u] \\
\mathrm{Map}(P^{n+1} C, A) \arrow[u, "(\ )\circ P^n(c)"] \arrow[r, shift left, "\mathrm{id}"] \arrow[r, shift right, "a\circ P(\ )\circ P^{n+1}(c)"'] & \mathrm{Map}(P^{n+1} C, A) \arrow[u, "(\ )\circ P^n(c)"']
\end{tikzcd}$$

(It is clear that it commutes, both for the identity maps and for the other horizontal maps.)

Now we want to calculate the limit of this diagram in two ways. First we calculate the limit of each column, yielding (4.3), and then we take equaliser of that, to obtain $\mathrm{Tw}(I, A)$. On the other hand, we can calculate the limit by taking first equaliser of each row. That gives in each row the space

$$\mathrm{Tw}(P^n C, A)$$

And then we can calculate the sequential limits of this new column. Now we should just note that all the maps in the first column are equivalences (well just bijections, if we work with hom sets). This follows from the Key Lemma. So the limit, which is what we are looking for, is just the zeroth space

$$\mathrm{Tw}(C, A)$$

as claimed. □

First we check some things:

(1) If f_n satisfies this condition, then f_{n+1} (equal to $a \circ P(f_n)$ does too. Indeed, apply P to the whole diagram, and paste on the bottom with an obvious square, to get

$$\begin{tikzcd}
I_{n+1} \arrow[r, "j_{n+1}"] \arrow[d, "Pf"'] & I_{n+2} \arrow[d, "P^2 f"] \arrow[dl, "Pf_{n+1}"] \\
PA \arrow[d, "a"'] & P^2 A \arrow[l, "Pa"] \arrow[d, "Pa"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

Here the left-hand composit is precisely f_{n+1}, and the right-hand composite is precisely $P(f_{n+1})$. The new diagonal (not drawn) is precisely $a \circ P(f_{n+1}) = f_{n+2}$. This is all as required.

(2) Assuming $n > 0$, we also have f_{n-1} defined as $f_n \circ j_{n-1}$. The claim is that f_{n-1} again satisfies the condition. We do have a diagram

$$\begin{tikzcd}
I_{n-1} \arrow[r, "j_{n-1}"] \arrow[dd, "f_{n-1}"'] & I_n \arrow[d, "j_n"] \arrow[ddl, "f_n"'] \\
 & I_{n+1} \arrow[d, "Pf_n"] \\
A & PA \arrow[l, "a"]
\end{tikzcd}$$

here the upper left-hand triangle commutes by definition of f_{n-1}, and the distorted square commutes by the condition on f_n. Finally the right-hand composite is $Pf_n \circ j_n = Pf_n \circ Pj_{n-1} = P(f_n \circ j_{n-1}) = P(f_{n-1})$. So indeed f_{n-1} satisfies the condition too.

Altogether the conclusion is that if we give just one $f_n : I_n \to A$ satisfying the condition, then there are uniquely given such maps for all n.

So we may as well just give $f_0 : C \to A$, which has to be a twisted morphism.

Reformulated a bit: to give just f_0 induces all the higher maps, by remark 1, and all the higher maps will then satisfy the condition. Conversely, if we have all maps f_n for all n, then in particular we have f_0, and could reconstruct all the other maps from that one. The final observation is that we could actually have started from any level map f_n, then we can extend both up and down.

There is missing here an argument that if we first extend down and then up, we get the map we started with. And if we first extend up and then down, we also get the map we started with.

4.6.15 Proposition. *To give a P-algebra homomorphism $(I_C, i) \to (A, a)$ is equivalent to giving a twisted morphism $(C, c) \to (A, a)$.*

We already know that the twisted morphism is the whole sequence of maps. It remains to see that the whole sequence of maps is precisely an algebra map.

Now we give up the assumption that P preserves filtered colimits. That really means ω-filtered colimits. Now assume instead that it preserves κ-filtered colimits. Here κ is a limit ordinal (= regular cardinal?). In this case, starting from a coalgebra (C, c) when taking the colimit $\operatorname{colim}_{n\in\omega} P^n C$, we do get a coalgebra again, but there is no guarantee the structure map is invertible. But then we can just apply P iteratively, and take the colimit again. We continue like this until we have a κ-long chain. The last step is taking colimit, because κ was assumed to be a limit ordinal. This time, since P preserves κ-filtered colimits, the resulting coalgebra will have invertibla structure map!

Hence it is also a P-algebra. Now the argument goes like before.

4.6.16 Theorem. $\operatorname{colim}_{n\leq\kappa} P^n C$ *is the initial P-algebra.*

The proof goes the same, except that the colimits and limits involved are longer than ω. That actually makes no difference for the argument. It does make some difference in notation: we need to be able to write $P^n C$ for ordinals bigger than ω. We just agree that it means to take the colimits involved in getting to that n...

Chapter 5

Polynomial monads and operads

Ch:I:operads

5.1 Polynomial monads

Cartesian monads

5.1.1 Monads. If $\mathscr{C}$ is a category (for example the category of sets), then there is a category $\mathrm{End}(\mathscr{C})$ whose objects are the endofunctors $\mathscr{C} \to \mathscr{C}$, and whose arrows are the natural transformations. The composition operator $\circ$ turns $\mathrm{End}(\mathscr{C})$ into a (strict) monoidal category; the unit object is the identity functor $\mathsf{Id}_{\mathscr{C}}$. By definition, a *monad* is a monoid in the monoidal category $(\mathrm{End}(\mathscr{C}), \circ, \mathsf{Id}_{\mathscr{C}})$, i.e. an endofunctor $P : \mathscr{C} \to \mathscr{C}$ equipped with natural transformations $\mu : P \circ P \Rightarrow P$ and $\eta : \mathsf{Id}_{\mathscr{C}} \Rightarrow P$, satisfying associativity and the unit law. So a monad is a triple (P, μ, η), but we will often refer to a monad just by the naming the functor part. In any case we will use the letters μ and η for the structure maps of any monad, so there is not much won in referring to them. . .

5.1.2 Cartesian monads. A monad (P, μ, η) is *cartesian* if P preserves cartesian squares and if μ and η are cartesian natural transformations. In this section (and perhaps throughout!) all our monads will be cartesian.

If S and T are two monads defined on the same category $\mathscr{C}$, then a monad map is a cartesian natural transformation $f : S \Rightarrow T$ making two obvious squares commute.

5.1.3 Polynomial monads. Let ***Poly*** denote the category of polynomial functors (in one variable) and their cartesian natural transformations, with monoidal structure given by composition of functors. A monoid in here is called a *polynomial monad*. In detail, a polynomial monad is a polynomial functor $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$ equipped with a composition law $\mu : P \circ P \to P$ with unit $\eta : \mathsf{Id} \to P$, satisfying the usual associativity and unit conditions; the structure maps μ and η should be cartesian natural transformations.

Note that polynomial functors always preserve cartesian squares (cf. 1.7.1).

5.1.4 Graphical interpretation. The composition law is described graphically as an operation of contracting trees (formal compositions of bouquets) to bouquets. We shall refer to B as the set of *operations*. Since we have a unit, we can furthermore think of E as the set of *partial operations*, i.e. operations all of whose inputs except one are fed with a unit.

5.1.5 The set map description of the composition law. It's about taking a height-2 tree and producing from it a bouquet (height-1 tree). To require the map $P \circ P \to P$ to be cartesian means that the fibres match. This means that this new bouquet must have the same set of leaves as the resulting height-1 tree. The cartesian condition is that there is a square

$$\begin{tikzcd}
P'(B) \times E \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d] \\
P(B) \arrow[r] & B
\end{tikzcd}$$

Here we use the compact differential notation, cf. 1.5.6.

5.1.6 Units. If a polynomial monad P is represented by $E \to B$, then the unit $\eta : \mathsf{Id} \Rightarrow P$ is represented by a cartesian square

$$\begin{tikzcd}
1 \arrow[r, "="] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[d] \\
E \arrow[r] & B
\end{tikzcd}$$

That is, the map $1 \to B$ singles out an element u in B (a distinguished operation) and the requirement that the morphism be cartesian means that the fibre over this element is singleton, so the operation u is unary, hence is represented by a bouquet with only one leaf. That's just what it means to have a cartesian morphism $\mathsf{Id} \to P$. Now there is furthermore the unit requirement, namely that grafting a big collection of units on any bouquet and then contracting is the same as doing nothing, and that if you graft any bouquet b on top of the unit bouquet, then you get b again. For this reason we draw the unit operation as a single edge without a dot.

5.1.7 Example: the pointed-set monad. The polynomial functor

$$\begin{array}{ccc} \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & 1+X \end{array}$$

has a natural monad structure, just like more generally the functor

$$X \mapsto E + X$$

for some fixed set E. Indeed, the natural maps

$$X \to E + X \leftarrow E + E + X$$

have to be the identity on the variable summand, and on the E-summand it is given by the unique monoid structure (with respect to $+$) that exists on any object. Exercise: this monad is cartesian.

5.1.8 Example: The free-monoid monad. The free-monoid functor

$$\begin{array}{ccc} \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & \sum_{n\in\mathbb{N}} X^n \end{array}$$

has a natural monad structure: the unit $\eta_X : X \to M(X)$ assigns the one-letter word (x) to each element in X, and the multiplication $\mu_X : M(M(X)) \to M(X)$ takes a word of words and concatenate them into a single word.

This monad is cartesian: the explicit graphical description makes is easy to check that μ and η are cartesian. In the case of the free-monoid monad: M itself has as operations the set of bouquets, and $M \circ M$ has as operations the set of two-level trees. The multiplication law μ just contracts inner edges to produce a bouquet from a two-level tree. Since it does not alter the number of input edges (the fibres), this is a cartesian map. Similar argument for η.

In contrast, the free-commutative-monoid monad is not cartesian. See Leinster's book. We are not so interested in this monad anyway since we know it is not polynomial. . .

5.1.9 Example: linear monads. Recall that a linear functor is a polynomial functor whose representing map is a bijection, and that we might as well assume it is an identity map. So a linear functor is given by just one set M, and it is

$$\begin{array}{ccc} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set} \\ X & \longmapsto & M \times X \end{array}$$

The composite of this functor with itself is then represented by $M \times M$, and it is easy to see that monad structure on this functor amounts precisely to monoid structure on the set M. Such linear monads are automatically cartesian. In conclusion, in the one-variable case, linear monads are just monoids. We shall see in the many-variable case **??** that linear monads are just small categories!

5.1.10 Partial composition law viewpoint. (The partial viewpoint for operads is advocated by Markl and Stasheff.) The composition law can be described in terms of partial operations as a map

$$B \times E \to B,$$

consisting in substituting one operation into one input of another operation.

The rest of this subsection is devoted to an analysis of the partial-composition viewpoint.

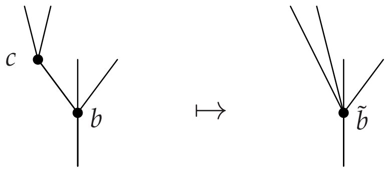
This viewpoint amounts to grafting only on one leaf. It is equivalent to the standard substitution because of the existence of units. We can define an operation by grafting onto only one leaf and then graft the unit on the remaining leaves. This defines an operation

$$P \times P' \to P$$

cf. the description we made in Number 1.5.9. To say that this is a cartesian operation (we have to check that cartesian composition law implies cartesian partial composition law, but that should be straightforward). That is to have a cartesian square

$$\begin{tikzcd}
E \times E + B \times (E \times_B E) \smallsetminus \Delta) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d] \\
B \times E \arrow[r] & B
\end{tikzcd}$$

The cartesian condition is that doing this contraction does not change the set of leaves! The important thing is that the fibre is the set of all leaves and that this does not change under the composition. This is the cartesian condition.



The picture (and the cartesian condition) indicates that the bouquet is obtained from the two-level tree by contracting dot c back to dot b. However, this $\tilde{b}$ cannot be the same as b because it has another fibre.

Here is the unit condition in terms of partial composition law: It's a morphism $1 \to B$ such that the induced map $E \to E \times B$ followed by composition $E \times B \to B$ is the projection itself:

$$\begin{tikzcd}
E \arrow[rr, "p"] \arrow[dr, "\eta"'] & & B \\
& E \times B \arrow[ur, "\mu"'] &
\end{tikzcd}$$

This is the expression of the requirement that grafting a single u on top on any bouquet at any leaf gives back the same bouquet.

The other requirement is that the section $1 \to B$ extends to $1 \to E$ (it does by the cartesian condition since there is only one point in the fibre over u) and that this is a section to both the map $s : E \to 1$ (which is obvious in the one-variable case) and also, as explained, is a section to $E \to B \to 1$. This section induces $B \to E \times B$ ($b \mapsto 1 \times b$) and this is a section to μ. That is, $B \to E \times B \to B$ is the identity map on B.

Exercise. Write out what associativity means in this viewpoint. There are two equations, corresponding to the fact that the top space over $B \times E$ has two components. One is the condition that it doesn't matter if you first graft on leaf e and then on another leaf f. The second condition is that it doesn't matter if you first graft g onto a leaf of f, and then graft the result onto a leaf of e, of if you first graft f onto e, and then graft g onto one of the leaves of f.

The first condition should amount to saying that some differential operator $P\partial^2$ is symmetric. The second condition is a differential equation looking something like

$$P\partial(P\partial(P)) \to P\partial(P) \quad = \quad (P\partial P)\partial P \to (P)\partial P$$

I haven't yet figured out precisely what it is. . .

The free monad on a polynomial endofunctor (one variable)

5.1.11 P**-trees.** Let P denote a polynomial endofunctor given by $E \to B$. We define a *P-tree* to be a tree whose nodes are decorated in B, and with the additional structure of a bijection for each node n (with decoration b) between the set of input edges of n and the fibre E_b.

Another description is useful: a B-tree is a tree with node set N, and node-with-marked-input-edge set N', together with a diagram

$$\begin{tikzcd}
N' \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & N \arrow[d, "\beta"] \\
E \arrow[r] & B
\end{tikzcd}$$

Then the β expresses the decorations, and the cartesian square encodes the bijections. As we saw when constructing them, the P-trees are obtained by freely grafting elements of B onto the leaves of elements of B, and formally adding a dotless tree. (We shall formalise this in 12.4.1.)

5.1.12 The free monad on a polynomial endofunctor. The adjunction

$$\begin{tikzcd}
P\text{-}\boldsymbol{Set} \arrow[d, shift left=2, "U"] \\
\boldsymbol{Set} \arrow[u, shift left=2, "F_P"] \arrow[u, phantom, "\dashv"]
\end{tikzcd}$$

described in 4.6.1 generates a monad $T_P := U \circ F : \boldsymbol{Set} \to \boldsymbol{Set}$. It is the *free monad* on P. We will denote it $\overline{P}$. We already described it: it sends a set X to the set of P-trees with leaves decorated in X. Hence it is represented by the map $\mathrm{tr}'(P) \to \mathrm{tr}(P)$ where $\mathrm{tr}(P)$ is the set of P-trees, and $\mathrm{tr}'(P)$ is the set of P-trees with one leaf marked.

The monad structure of $\overline{P}$ is described explicitly in terms of grafting of trees. In a partial-composition description, the composition law is

$$\mathrm{tr}(P) \times \mathrm{tr}'(P) \to \mathrm{tr}(P)$$

consisting in grafting a tree onto the specified input leaf of another tree. The unit is given by $1 \to \mathrm{tr}(P)$ singling out the dotless tree.

A better description: nodes are decorated in B, and the additional structure of a bijection between the set of input edges of node n with decoration b and the fibre E_b.

5.1.13 Lemma. *The free monad on a polynomial functor is cartesian.*

Proof. We must show that $\mu : \overline{P} \circ \overline{P} \to \overline{P}$ and $\eta : \mathsf{Id} \to \overline{P}$ are cartesian. let us start with η since it is the easiest: the map $1 \to \mathrm{tr}(P)$ singles out the dotless tree. We need to show that the square

$$\begin{tikzcd}
\mathrm{tr}'(P) \arrow[r] & \mathrm{tr}(P) \\
1 \arrow[u] \arrow[r] & 1 \arrow[u]
\end{tikzcd}$$

is cartesian, but this is true because the dotless tree has precisely one leaf.

We will apply the same argument to the composition map: intuitively, the reason the composition is cartesian is that a two-level tree and the one-level tree obtained by the composition map has the same number of leaves (and the same decorations). (Note that this is the same argument as for the unit, where we observed that both had a singleton fibre).

But let us work through the involved sets: the base set for the composite polynomial functor is $p_*(E \times B)$, which we identified as the set of trees whose leaves are decorated by trees. This is the same thing as a tree with a cross section. (Note that in this interpretation we are implicitly using the fact that we have units: a cross section might cross a leaf, in which case we need to interpret the upper part of that leaf as dotless tree, which we can do because we have formally added the dotless trees.) The top set of $\overline{P} \circ \overline{P}$ is the set of trees with a cross section and one leaf marked. The composition map consists in forgetting the cross section. Clearly this operation does not change the number or decoration of the leaves, so we see that fibres for q and p are in natural bijection. That's all. □

I wanted to understand the cartesian condition also in the partial substitution viewpoint. . .

5.1.14 Alternative fixpoint construction. There is an alternative description of the free monad, which we briefly mention: I HAVE NOT YET FIGURED OUT THE RELATIONSHIP BETWEEN THE TWO FIXPOINT CONSTRUCTIONS. The free monad is a least fixpoint for the functor

$$\begin{aligned} \Gamma := \Gamma_P : \boldsymbol{Poly}^{\mathrm{c}} &\longrightarrow \boldsymbol{Poly}^{\mathrm{c}} \\ Q &\longmapsto \mathsf{Id} + P \circ Q. \end{aligned}$$

One can check that Γ preserves monos, hence there is a sequence of monos

$$\mathsf{Id} \hookrightarrow \Gamma(\mathsf{Id}) \hookrightarrow \Gamma^2(\mathsf{Id}) \hookrightarrow \dots$$

If P is finitary (i.e., $p : E \to B$ is a finite map—this is the case in most our examples) then it preserves such sequential colimits, and the least fixpoint can be constructed as

$$\overline{P} = \bigcup_{n \geq 0} \Gamma^n(\mathsf{Id}).$$

The equation satisfied by $\overline{P}$,

$$\overline{P} = \mathsf{Id} + P \circ \overline{P},$$

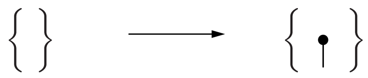
expresses the recursive characterisation of trees: a tree is either a dotless tree or a finite set of trees.

Otherwise we need transfinite induction. . .

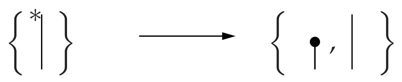
Examples

5.1.15 Example. The free-monoid monad is not a free monad on anything.

5.1.16 Example. (The pointed-set monad.) The pointed-set monad is the free monad on the constant polynomial functor $X \mapsto 1$. The functor $X \mapsto 1$ is represented by the set map



The free monad on this is represented by



$$X \mapsto 1 + X$$

the pointed-set monad.

5.1.17 Example. Similarly, for a fixed set S, we can consider the constant polynomial functor $X \mapsto S$. The free monad on this one is

$$X \mapsto S + X.$$

A special case of this situation is the constant polynomial functor $X \mapsto \emptyset$. It generates the identity monad $X \mapsto X$.

5.1.18 Example. The free monad on id : ***Set*** $\rightarrow$ ***Set***, is the free $\mathbb{N}$-action monad $X \mapsto \mathbb{N} \times X$. (Its Eilenberg-Moore algebras are the $\mathbb{N}$-sets, i.e. set with an $\mathbb{N}$-action.)

5.1.19 Example. Construct the free monad on $2 \rightarrow 1$. It is going to be the monad whose operations are the binary trees.

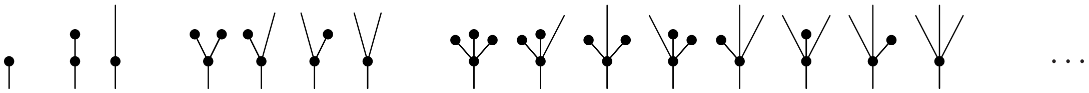
(Note that if we don't include a nullary operation, then after the first iteration we have only the dotless tree as operation. Then in the next step we get only the binary bouquet, plus the dotless tree. And then we can graft these onto the binary bouquet, and in the end we get all binary trees (planar) but not levelled. If we include a nullary operation, then we also allow stopdots in the otherwise binary trees. . .)

No it seems we don't get levelled trees. The reason is that in each step of the iteration, the base bouquet is from the original P. This means in a sense that every tree is obtained only with one parenthesisation, namely so-to-speak with all the parentheses near the root,

5.1.20 The free monad on M: planar trees. The really interesting example to work out is of course when we start with M, the free-monoid monad.

The first iteration is about the composite $M \circ 0$. The operations for this composite endofunctor is the set of ways to decorate bouquets in $\emptyset$. Only the nullary operation survives as operation generated in this way, and then we add the dotless tree, to account for adding Id. Now we have a polynomial endofunctor T with two operations: the stopdot and the dotless tree.

In next iteration we compute $M \circ T$: its operations are obtained by decorating bouquets with the dotless tree or the stopdot. We get



and then add the dotless tree again.

And so on. It is clear that in the limit we get exactly the planar rooted trees with boundary. These are M-trees.

Planarity is just one way to encode that for each node there is a specified bijection between the input edges and the corresponding fibre of the map $\mathbb{N}' \to \mathbb{N}$.

5.2 Classical definition of operads

An *operad* (more precisely, a *non-symmetric* or *planar* operad) consists of a sequence of sets $(A_n \mid n \in \mathbb{N})$ equipped with a many-in/one-out composition law: for each $n, k_1, \ldots, k_n$ (natural numbers) a map

$$\begin{array}{rcl} A_n \times A_{k_1} \times \cdots \times A_{k_n} & \longrightarrow & A_{k_1+\cdots+k_n} \\ (b; a_1, \ldots, a_n) & \longmapsto & b \circ (a_1, \ldots, a_n) \end{array}$$

and a specified identity operation $1 \in A_1$ satisfying associativity and unit axioms:

$$\begin{aligned}\big(b \circ (a_1, \ldots, a_n)\big) \circ (a_{1,1}, \ldots, a_{1,m_1}, a_{2,1}, \ldots, a_{2,m_2}, \ldots, \ldots a_{n,1}, \ldots, a_{n,m_n}) \\ = b \circ \big(a_1 \circ (a_{1,1}, \ldots, a_{1,m_1}), \ldots, a_n \circ (a_{n,1}, \ldots, a_{n,m_n})\big)\end{aligned}$$

$1 \circ a = a$ and $b \circ (1, \ldots, 1) = b$.

The elements of A_n are called n-ary operations.

We picture an element $b \in A_n$ as a bouquet with n input leaves (and label the dot with the name of the element). The composition law then consists in grafting the sequence of operations $(a_1, \ldots, a_n)$ onto the leaves. Note that these a_i form an ordered sequence, so in order to know which leaf we are grafting each of them onto, we need to require that the drawing is planar, in contrast to our convention for the bouquets for polynomial functors.

With the graphical interpretation, composition and the associativity axiom are much easier to understand: given a bottom operation b with n leaves, and n operations $a_1, \ldots, a_n$, and for each input leaf of each of these another operation $a_{i,j}$, then it doesn't matter whether we first graft the a_i onto b, and then finally graft the long list of $a_{i,j}$ onto the union of all the leaves of the a_i, OR if we first take each a_i and fill all its leaves with the operations $a_{i,1}, \ldots, a_{i,m_i}$ and then graft the resulting n trees on top of b.

A *morphism of operads*, say from A to B, is a sequence of maps $f = (f_n : A_n \to B_n \mid n \in \mathbb{N})$ such that and $f_1(1) = 1$, and $f_{k_1+\cdots+k_n}(b \circ (a_1, \ldots, a_n)) = f_n(b) \circ (f_{k_1}(a_1), \ldots, f_{k_n}(a_n))$. (Here is it understood that $a_i \in P_{k_i}$.)

5.2.1 Partial composition laws. Since we have a unit, we can make sense of partial composition laws: there are k partial composition laws

$$\begin{array}{rcl} A_n \times A_k & \longrightarrow & A_{n-1+k} \\ (b; a) & \longmapsto & b \circ_i a \end{array}$$

consisting in grafting a onto the ith leaf of b and grafting units onto the remaining $n-1$ leaves.

WRITE THE AXIOMS FOR THIS VIEWPOINT

5.3 The monoidal category of collections

By a *collection* we mean a $\mathbb{N}$-graded set, i.e. a set A equipped with a map to $\mathbb{N}$. Hence we can write

$$A = \sum_{n \in \mathbb{N}} A_n.$$

A map of collections is just a set map compatible with the grading, i.e. a morphism in the slice category $\boldsymbol{Set}/\mathbb{N}$. In other words, the category of collections is

$$\boldsymbol{Coll} := \boldsymbol{Set}/\mathbb{N}.$$

A collection is just like an operad but without the extra structure of the composition law and the unit law. So we will still think of the elements of A_n as n-ary operations, and we picture them as planar bouquets with n input leaves.

We now construct the substitutional tensor product of collections. It is non-symmetric. Namely given two collections A and B, we define a set $B \otimes A$ as the set of all ways of sticking operations from A into operations in B

AGREE ON A CONVENTION FOR COMPOSITION! SHOULD WE USE CONVENTIONAL $\circ$ NOTATION? the convention is currently dictated by compatibility with composition of endofunctors, written in standard $\circ$ backwards notation.

Since the inputs of an element in B are ordered, and there are n of them for some $n \in \mathbb{N}$, we need a list of n elements from A. Altogether we have

$$B_n \times A^n$$

We do this for each $n \in \mathbb{N}$, so the total set of $B \otimes A$ is

$$B \otimes A := \sum_{n \in \mathbb{N}} B_n \times A^n.$$

Now we need to specify how this set is $\mathbb{N}$-graded, i.e. give it structure of a collection. The idea is that the degree is the total number of input leaves. In other words,

$$\deg(b; a_1, \ldots, a_n) := \sum_{i=1}^{n} \deg(a_i)$$

So if we want to describe the set $B \otimes A$ degree-wise, the degree-n piece is given by the formula

$$\sum_{d \in \mathbb{N}} \sum_{i} B_d \times A_{i_1} \times A_{i_2} \times \cdots \times A_{i_d}$$

the inner sum is over all ordered partitions of the integer n into d parts,

$$i_1 + \ldots i_d = n$$

Check the axioms for a monoidal category. To see that it is unital is not so bad: the collection $1 \to \mathbb{N}$ (which picks the one-element in $\mathbb{N}$) is a unit. Indeed, this collection has only one operation, and it is of degree 1, so the ugly sums reduce to a single term. In the graphical viewpoint this is easy to see: for each $b \in B_n$ there is precisely one way to graft the unique operation of 1 onto the leaves of b, so $1 \otimes B = B$. Conversely, if we want to graft all possible operations of A onto the unique operation of 1, we get A again.

5.3.1 ***M*-collections.** Collections take sequences of elements as input. For this reason we can formulate the notions in terms of the free-monoid monad, and it becomes much easier to handle. So the notion of collection is really M-collection, and later on we shall see how any cartesian monad P (and polynomial ones in particular) gives rise to a notion of P-collection and P-operad (generalised operad).

If A and B are collections, the substitutional tensor product is defined as

$$B \otimes A := B \times_{\mathbb{N}} M(A) = \sum_{n \in \mathbb{N}} B_n \times A^n.$$

The map $M(A) \to \mathbb{N}$ is simply $M(A) \to M(1) = \mathbb{N}$, obtained by taking M on the unique map to the singleton set. The fibre-product condition simply says that we take an n-tuple of elements in A when B is of arity n. The unit is $1 \to \mathbb{N}$, which is precisely η_1, the unit for the monad M.

The degree map from $B \times_{\mathbb{N}} M(A)$ to $\mathbb{N}$ is not just the map via the fibre condition (that would amount to letting the tensor product always have the same degree as its second factor). It is defined using the monad structure on M. Namely first project onto $M(A)$, then take the map M on the structure map $A \to \mathbb{N} = M(1)$; this gives $M(A) \to M(M(1))$ and then apply the multiplication map μ_1, which is nothing but addition in $\mathbb{N}$. This degree map agrees with the original description of the degree: from the sequence of elements in A we first get a sequence of natural numbers, and then we add them. (So the degree is the total number of leaves of the operations listed.)

Now it is easy to check the unit law: note that in order to check that $1 \times_{\mathbb{N}} M(A)$ is isomorphic to A, we use the cartesian condition on η: we have

$$\begin{tikzcd}
A \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[d] \\
M(A) \arrow[r] & M(1)
\end{tikzcd}$$

(So on one side of the unit axiom verification, we used only monoid structure on $\mathbb{N}$, not the general multiplication map of M. On the other side we used that η is cartesian.)

Now verify the associativity.

All this is just an easy case of Burroni's construction.

5.3.2 Proposition. *An operad is the same thing as a monoid in* $(\boldsymbol{Coll}, \otimes, 1)$.

5.3.3 M**-operads.** The operads we have considered are really M-operads. They can also be described as collections $X/\mathbb{N}$ equipped with an action

$$X \times_{\mathbb{N}} M(X) \to X$$

which must be associative and unital.

5.4 Finitary polynomial functors and collections

Definition. A map $p : E \to B$ is called finite if every fibre is finite. A polynomial functor is called *finitary* if represented by a finite map. Let ***FinPoly*** denote the category of finitary polynomial functors and cartesian natural transformations.

Let P be a finitary polynomial functor, represented by the finite map $p : E \to B$. Then the classifying map (1.3.2)

$$\begin{array}{ccc} B & \longrightarrow & \mathbb{N} \\ b & \longmapsto & |E_b| \end{array}$$

gives us already a collection. Clearly this is functorial and gives us a functor ***FinPoly*** $\to$ ***Coll***.

In the other direction there is also an obvious functor: given a collection $B/\mathbb{N}$ we can form the pullback square

$$\begin{tikzcd}
\mathbb{N}' \times_{\mathbb{N}} B \arrow[r] \arrow[d] & B \arrow[d] \\
\mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd} \tag{5.1}$$

polyfromcoll

the the top row represents a polynomial functor (clearly finitary).

If we start with a collection, constructs a finitary polynomial functor, then the classifying map is exactly the collection we started with. If we start with a finitary polynomial functor, and consider the polynomial functors constructed from the classifying map, then it is clear that we get something isomorphic. It is an easy mistake to make to think that this pair of functors constitute an equivalence of categories between ***FinPoly*** and ***Coll***. But it is certainly not an equivalence: closer inspection reveals that the functor ***FinPoly*** $\to$ ***Coll*** is not even faithful. Indeed, the finitary polynomial functor M is clearly sent to the terminal collection $\mathbb{N}/\mathbb{N}$ but M has many automorphisms!

(The situation is similar to the case of a functor from a nontrivial group G (considered as a one-object category) to the terminal category. There is an obvious (in fact unique) functor in the opposite direction, but clearly this is not an equivalence.)

The issue is that although the classifying map is uniquely given, its extension to a cartesian square

$$\begin{tikzcd}
E \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[d, "\kappa"] \\
\mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

is *not* unique, as we have already observed (2.1.9).

So every finitary polynomial functor admits a cartesian map to the free-monoid monad M, but this map is not unique so M is *not* a terminal object in ***Poly***$^{\text{c}}$.

In order to get an equivalence of categories, we need to retain more information in the passage from collection to polynomial functor: instead of just returning the top row of the diagram (5.1) we retain the whole diagram, so as to land in the category $\boldsymbol{Poly}^{\mathrm{c}}/M$ of polynomial functors over M (necessarily finitary since M is).

This functor is an equivalence.

5.4.1 Proposition. *The two functors just described constitute an equivalence of categories*

$$\boldsymbol{Poly}^{c}/M \simeq \boldsymbol{Coll}.$$

We will now show that this is furthermore an equivalence of monoidal categories, where the monoidal structure on $\boldsymbol{Poly}^{\mathrm{c}}/M$ is composition, and on ***Coll*** the substitutional tensor product for collections. Thereby we also establish an equivalence of categories between monads over M and operads.

Equivalence of monoidal categories

MENTION CLUBS, AND REFER TO [58]

Let us look at polynomial functors with a specified cartesian map to M (the free-monoid endofunctor, represented by $\mathbb{N}' \to \mathbb{N}$). (Note that having such a cartesian map implies that P is finitary.)

The category of polynomial functors with a (cartesian) morphism to M is monoidal: given $P \to M$ and $Q \to M$, their composite $P \circ Q$ has a canonical map to M using the monad structure on M: namely

$$P \circ Q \to M \circ M \xrightarrow{\mu} M$$

The unit is he identity endofunctor with structure map $\eta : \mathsf{Id} \Rightarrow M$ (the unit for the monad M).

Note that $P \circ Q \Rightarrow M \circ M$ is a cartesian natural transformations: it is the horizontal composite of two cartesian natural transformations, and since all the involved functors preserve pullbacks, this is again a cartesian natural transformation. Cf. 2.5.2.

5.4.2 Proposition. *There is an equivalence of monoidal categories between the category* $(\boldsymbol{Poly}^{c}/M, \circ, \mathrm{id})$ *and the category of collections* $(\boldsymbol{Coll}, \otimes, 1)$.

We already established the equivalence of categories. . . Now use the notion of M-collection to see easily that this equivalence is monoidal.

(Note that the terminal object in the first category is M, corresponding to the terminal collection $\mathbb{N} \to \mathbb{N}$. A terminal object in a monoidal category always carries a unique monoid structure: the polynomial functor M is in fact a monad, and the terminal collection is in fact an operad.)

Note that a monoid in the monoidal category $\boldsymbol{Poly}^{\mathrm{c}}/M$ is the same as an object in the slice category $\boldsymbol{PolyMnd}/M$. These in turn by the Proposition correspond to monoids in $\boldsymbol{Coll}$ and these are precisely the operads:

Mmonad=operad **5.4.3 Corollary.** *There is an equivalence of categories*

$$\boldsymbol{PolyMnd}/M \simeq \boldsymbol{Opd}.$$

There is also the forgetful functor $\boldsymbol{Poly}^{\mathrm{c}}/M \to \boldsymbol{Poly}^{\mathrm{c}}$. Composing with the equivalence $\boldsymbol{Coll} \to \boldsymbol{Poly}^{\mathrm{c}}/M$ we get a functor $\boldsymbol{Coll} \to \boldsymbol{Poly}^{\mathrm{c}}$: it is simply the map that takes a collection $B \to \mathbb{N}$, and returns the polynomial functor $\mathbb{N}' \times_{\mathbb{N}} B \to B$ (the pullback of the universal family).

5.5 The free operad on a collection

free-operad **5.5.1 The free-operad monad.** The forgetful functor from $\boldsymbol{Opd}$ to $\boldsymbol{Coll}$ has a left adjoint F, which to any collection associates the operad obtained by freely combining all the operations in all possible ways, and formally adding a unit (a dotless tree). So the free operad on A is the set T of trees with each node of degree n decorated by an element in A_n. The grading of T is given by the number of leaves.

It is just the general construction of the free monoid on an object in a monoidal category. This might not always exist, but it does in this case since it is just the free monad on a polynomial functor.

We will later come to polynomial functors in many variables. An important example are polynomial functors in countably many variables: these are functors $\boldsymbol{Set}/\mathbb{N} \to \boldsymbol{Set}/\mathbb{N}$, and we shall see that the free-operad endofunctor on $\boldsymbol{Set}/\mathbb{N}$ (which is a monad since it was generated by an adjunction) is polynomial.

constellations **5.5.2 Example.** We have now understood the free-operad monad $F : \boldsymbol{Coll} \to \boldsymbol{Coll}$. Now we can study its Lambek algebras. So define a F-collection to be a collection $X/\mathbb{N}$ equipped with a map of collections $FX \to X$. Now it turns out the forgetful functor F-***Coll*** $\to$ ***Coll*** has a left adjoint C, the free F-collection on a collection.

Figure out what it is. We already observed that the free operad on a collection is the collection of planar trees built from the operations of the original collection. Figure out that the free F-collection on a collection X is the set of X-trees with circles! I.e. planar constellations.

Thinking of $F(1)$ as being given by a set of planar bouquets, $C(1)$ is the set of trees of trees. These are planar constellations. I.e. circles set in planar trees. Since F is a monad, there is for each collection X a natural map of collections $C(X) \to F(X)$. In particular, for the terminal collection 1: a map associating to a constellation a tree: it is just to erase the circles.

5.6 P-operads

See Leinster [75] Section 4.2.

The constructions and results of the previous section work with any polynomial monad in the place of M, giving rise to notions of P-collection and P-operad. We shall not go too much into detail here, leaving a more thorough treatment to the many-variable case in Chapter 13.

Let $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$ be a polynomial functor, represented by $E \to B$. The category of polynomial functors over P, denoted $\textbf{\textit{Poly}}^{\mathrm{c}}/P$, is naturally a monoidal category under composition: the composite of $X \to P$ and $Y \to P$ is the composite endofunctor $X \circ Y$ equipped with a map to P defined by $X \circ Y \to P \circ P \to P$ (cartesian by 2.5.2). The unit for the monoidal structure is the identity functor, equipped with structure map $\eta : \mathsf{Id} \Rightarrow P$.

Define the category of P-collections as $\textbf{\textit{Coll}} = \textbf{\textit{Set}}/P(1)$. Thereby there is induced a tensor product of P-collections: it is not difficult to check that this tensor product is given by $B \otimes A := B \times_{P(1)} P(A)$. The unit is $\eta_1 : 1 \to P(1)$.

There is an equivalence of monoidal categories

$$\begin{array}{rcl} \textbf{\textit{Poly}}^{\mathrm{c}}/P & \longrightarrow & P\text{-}\textbf{\textit{Coll}} \\ X & \longmapsto & X(1) \end{array}$$

The inverse equivalence takes a P-collection $A \to P(1)$ and returns the polynomial functor represented by $E \times_B A$.

In fact there is no need to use polynomial monads for this to work: it works with cartesian monads in general. If doing it in this generality, we should just notice that if P is polynomial, then all the P-collections correspond to polynomial functors again. (Recall from 2.5.1 that any functor with a cartesian natural transformation into a polynomial one is again polynomial.

Next, since equivalent monoidal categories have equivalent categories of monoids, we see that polynomial monads over P correspond to P-operads (we define P-operads in this way). This is a very well-known and basic result. The only novelty is that if P is a polynomial monad, then all P-operads are 'polynomial'. In particular the free P-operad is the same thing as the free monad over P:

$$\begin{tikzcd}
\textbf{\textit{PolyMnd}}/P \arrow[d, shift left] \arrow[d, phantom, "\dashv"] & P\textbf{\textit{-Opd}} \arrow[d, shift left] \arrow[d, phantom, "\dashv"] \\
\textbf{\textit{Poly}}^{\mathrm{c}}/P \arrow[u, shift left] \arrow[r, phantom, "\simeq"] & \textbf{\textit{Set}}/P1. \arrow[u, shift left]
\end{tikzcd}$$

NOT SURE WHAT I MEANT HERE:

In fact we have an equivalence of categories between P-**_alg_** and the category of algebras for the free operad on the terminal collection. The free operad on the terminal collection is just $T1 \to P1$. Its algebras are understood to be the Eilenberg-Moore algebras for the corresponding monad which is none other than T.

But we know that there is a canonical equivalence of categories (the Eilenberg-Moore comparison functor)

$$P\textbf{\textit{-alg}} \xrightarrow{\Phi} \textbf{\textit{Set}}^T$$

Cf. comments earlier in the text, quoting [13, Ch.9].

Chapter 6

[Polynomial functors in computer science]

ch:computing

The category of sets is not a good model for type theory and computer science. They rather like to work in some constructive setting like for example the effective topos, a topos with the feature that every function is recursive. Type theory takes place in a locally cartesian closed category. . .

PERHAPS THIS WHOLE CHAPTER SHOULD BE MOVED TO THE MANY-VARIABLE PART

Copy from Manes–Arbib [79].

Cf. Dybjer, Abbott, Altenkirch, etc.

Abbott, *Categories of containers* [1], Abbott–Altenkirch–Ghani [2], [3], [4], Abbott–Altenkirch–Ghani–McBride [5], [6], Altenkirch–Morris [8]

6.1 Data types

6.1.1 Type polymorphism. The standard example of a polymorphic data type is `list`, which is the generic notion which can be instantiated to lists of integers, lists of booleans, lists of pairs of integers, lists of lists of integers, etc. From a categorical viewpoint, the list constructor is a functor

$$\mathbf{\mathit{Type}} \to \mathbf{\mathit{Type}},$$

associating to a given data type, the new data type of lists of the given type. More generally, *polymorphic data types* are endofunctors of ***Type***. Here, for the sake of simpification, we will pretend that the category ***Type*** is the category of sets.

The operations that can be applied to polymorphic data types independently of their instantiation, called *polymorphic functions*, are precisely natural transformation.

An example of a polymorphic data type is the type constructor `5-tuple` (i.e. list of length 5): it makes sense for any given type. Another example is `4-tuple`. An example of a polymorphic function is the operation of truncating a 5-tuple to a 4-tuple by discarding the last element in the 5-tuple. Clearly this operation makes sense independently of what it is a 5-tuple of.

An example of an operation which is not polymorphic function is transforming a 4-tuple into a 5-tuple by appending a zero. This function only makes sense for 4-tuples of something where zero makes sense, like 4-tuples of integers. It does not a priori make sense for 4-tuples of fruits: if you append a zero to a 4-tuple of fruits you do not get a 5-tuple or fruits. Another example of a function that it not polymorphic is the operation of sorting a list of integers according to size. Of course, just as in the previous non-example, this does not make sense for all base types, but furthermore the operation is not independent of the data.

Let us pretend that types are sets. Then the data type constructor 5-tuple is the functor

$$\begin{array}{rcl} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set} \\ X & \longmapsto & X^5 \end{array}$$

Similarly, the 4-tuple constructor is $X \mapsto X^4$. The truncate function should now be a natural transformation from X^5 to X^4. Clearly it's component at X is just projecting away the last factor of the five-fold product. We can find the polynomial representation of this polymorphic function: the type `5-tuple` is represented by the polynomial (map of sets) $5 \to 1$ and `4-tuple` is represented by the map $4 \to 1$. The natural inclusion

$$4 \hookrightarrow 5$$

(omitting the last element in 5) represents the natural transformation which is therefore of representable type. Note that this natural transformation is not cartesian. So in data type theory, it is important not to restrict attention to cartesian natural transformations.

6.1.2 Historical remarks. Notions of type polymorphism and type constructors go back at least 40 years, Strachey [99] (subsequent milestones include Girard [42] and Reynolds [94]) and is now a built-in feature of many modern programming languages such as ML and Haskell.

The idea of containers, whose prehistory is also rather long, is that most interesting polymorphic data types, inductive types in particular, can be analysed in terms of their shape: such types are given by a 'set' of shapes A, and for each $a \in A$ a 'set' B_a of positions of shape a. For example, the list type has $\mathbb{N}$ as set of shapes (the shape of a list is just its length), and for each $n \in \mathbb{N}$ the set of positions is $\mathbf{n} = \{0, 1, \ldots, n-1\}$. Instantiating in a set X amounts to mapping the positions into X, hence the functor is

$$X \mapsto \sum_{n \in \mathbb{N}} X^{\mathbf{n}} .$$

6.1.3 Append. An important operation on a list consists in appending some element to it. In other words, given a list and an extra element (of the same type), you can build a new list, one longer than the original, by putting the extra element at the end of the list. More formally, for a fixed type X, the list and the extra element together is an element in

$$L(X) \times X$$

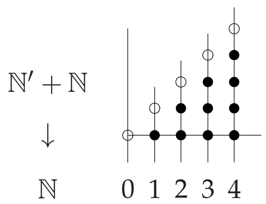
and we want to get from this an element in $L(X)$. So we are speaking about a natural transformation

$$L \times \mathsf{Id} \Rightarrow L.$$

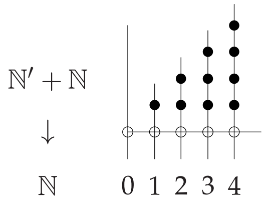
Now L is represented by the map $\mathbb{N}' \to \mathbb{N}$ as usual. On the other hand, $L \times \mathsf{Id}$ is represented by $\mathbb{N}' + \mathbb{N} \to \mathbb{N}$ (by the rule for the product of two polynomial functors). The natural transformation is given by

$$\begin{tikzcd}
L \times \mathsf{Id} : & \mathbb{N}' + \mathbb{N} \arrow[r] \arrow[d, "g"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N} \arrow[d, "+1"] \\
L : & \mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

The fact that the base map is $+1$ simply says that the list becomes one longer than the original list. The other map g is where we specify how this prolongation is done: to define the map $\mathbb{N}' + \mathbb{N} \to \mathbb{N}'$ we need on the first summand to send (n, i) to some $(n+1, j)$ (in order for the diagram to commute). There are two ways of defining j uniformly: we can take $j := i$ or we can take $j := i + 1$. Since we want the square to be cartesian, we need to have bijection on the fibres, so if we choose $j := i$, then on the summand $\mathbb{N} \to \mathbb{N}'$ we must send the unique point $n \in \mathbb{N}$ to $(n, n-1) \in \mathbb{N}'$, the unique point over n not hit by the first summand. Here is a picture of this way of fitting $\mathbb{N}' + \mathbb{N}$ into $\mathbb{N}'$, with the images of the elements of $\mathbb{N}$ drawn as white dots and the images of the elements of $\mathbb{N}'$ drawn as black dots:



On the other hand, if we choose $j := i + 1$ then the second summand should always map to the bottom, i.e. $\mathbb{N} \to \mathbb{N}'$ mapping n to $(0, n)$. Here is a drawing of this way of fitting $\mathbb{N}' + \mathbb{N}$ into $\mathbb{N}'$:



These two options correspond to the the two different natural transformations: prepending and appending. So we find that both of these polymorphic functions are cartesian natural transformations.

6.1.4 Concatenation. Given two lists W_1 and W_2, we can obtain a single list by concatenation. If W_1 has length n_1 and W_2 has length n_2, then the result will have length $n_1 + n_2$. Formally we are describing a natural transformation from

$$L \times L \Rightarrow L.$$

The data type constructor $L \times L$ of pairs of lists is represented by the family

$$\mathbb{N}' \times \mathbb{N} + \mathbb{N} \times \mathbb{N}' \longrightarrow \mathbb{N} \times \mathbb{N}.$$

For a fixed size $(n_1, n_2) \in \mathbb{N} \times \mathbb{N}$ of a pair of lists, the fibre is of cardinality $n_1 + n_2$. This is also the size of of the result of the concatenation, so we see that the natural transformation is going to be cartesian. Hence a cartesian square

$$\begin{tikzcd}
L \times L : & \mathbb{N}' \times \mathbb{N} + \mathbb{N} \times \mathbb{N}' \arrow[r] \arrow[d, "g"'] & \mathbb{N} \times \mathbb{N} \arrow[d, "+"] \\
L : & \mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

Again it is the defition of the map g that really pinpoints the transformation. On the left-hand summand, $\mathbb{N}' \times \mathbb{N}$ we must map an element $((n_1, i_1), n_2)$ to some $(n_1 + n_2, j)$, and since we are talking about an i_1 pointing into the the first factor, we should put $j := i_1$. On the right-hand summand, $\mathbb{N} \times \mathbb{N}'$ we must map an element $(n_1, (n_2, i_2))$ to some $(n_1 + n_2, j)$ and we should now take $j := n_1 + i_2$, so as to fill up the fibre.

6.1.5 An example of a zip(?). Consider the data type constructor R of rectangular arrays of size $(p, q) \in \mathbb{N} \times \mathbb{N}$. (Note that this involves more than one different type of empty array, e.g. of size $(7, 0)$, $(3, 0)$, or $(0, 4)$.) As an endofunctor of ***Set*** it is represented by

$$\mathbb{N}' \times \mathbb{N}' \longrightarrow \mathbb{N} \times \mathbb{N}.$$

(Note that this is not the product in the category of polynomial functors.) The zip map takes all the rows of the array and concatenate them into a single list, so we are describing a natural transformation $R \Rightarrow L$. So if the array is of size (p, q), the resulting list will have length pq. (Note that p is the number of rows, and each row has length q.) As in the previous examples, we see that this natural transformation is going to be cartesian. It is given by

$$\begin{tikzcd}
R : & \mathbb{N}' \times \mathbb{N}' \arrow[r] \arrow[d, "g"'] & \mathbb{N} \times \mathbb{N} \arrow[d, "\times"] \\
L : & \mathbb{N}' \arrow[r] & \mathbb{N}.
\end{tikzcd}$$

The map g is defined as

$$g((p, i), (q, j)) = (pq, iq + j).$$

Of course we could define another zip, by concatenating all the columns of the array. It would be defined by $g((p,i),(q,j)) = (pq, jp+i)$.

We may also notice that each of these two natural transformation has two natural sections. One consists in interpreting a list as a 1-row array; this is the natural transformation $L \Rightarrow R$ given by $n \mapsto (1,n)$ on the base level, and $(n,i) \mapsto ((1,0),(n,i))$. The other consists in interpreting a list as a 1-column array; this is the the natural transformation $L \Rightarrow R$ given by $(n,i) \mapsto ((n,i),(1,0))$.

6.1.6 Some other operations on lists. *Repeat last element (if there is one).* So this operation sends a non-empty list $(x_0,\ldots,x_n)$ to $(x_0,\ldots,x_n,x_n)$. We are dealing with a natural transformation $L \Rightarrow L$ Since the length incremented by one (except for the empty list), on the base sets the map is $\ell(n) = n+1$ for $n > 0$ and $\ell(0) = 0$. The whole diagram is

$$\begin{tikzcd}
L: & \mathbb{N}' \arrow[r] & \mathbb{N} \\
 & K \arrow[u, "s"] \arrow[r] \arrow[d, "d"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N} \arrow[u, equal] \arrow[d, "\ell"] \\
L: & \mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

Here $K = \{(n,i) \in \mathbb{N} \times \mathbb{N} \mid n \neq 0, i \leq n\}$, and the map d sends (n,i) to $(n+1,i)$, and the other map is

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

Tail. This operation removes the first element (if there is one). This shortens the list by one, so the 'length map' $\ell : \mathbb{N} \to \mathbb{N}$ is

$$\begin{array}{ccc} \mathbb{N} & \longrightarrow & \mathbb{N} \\ n & \longmapsto & \begin{cases} n-1 & \text{if } n > 0 \\ 0 & \text{if } n = 0 \end{cases} \end{array}$$

The whole diagram is

$$\begin{tikzcd}
L : & \mathbb{N}' \arrow[r] & \mathbb{N} \\
 & K \arrow[u, "s"] \arrow[r] \arrow[d, "d"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N} \arrow[u, equal] \arrow[d, "\ell"] \\
L : & \mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

In this case $K = \{(n,i) \mid 0 < i < n\}$, and s is the evident embedding $(n,i) \mapsto (n,i)$. On the other hand, $d(n,i) = (n-1, i-1)$.

Pop. This one is nearly the same: remove the last element if there is one.

Truncate to length k. This operation sends a list $(x_0, \ldots, x_k, \ldots, x_n)$ to $(x_0, \ldots, x_k)$ if $n > k$, and it does nothing on shorter lists.

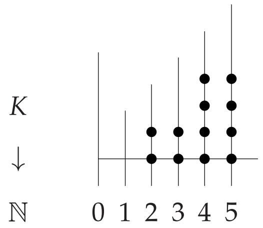
$$\begin{tikzcd}
L : & \mathbb{N}' \arrow[r] & \mathbb{N} \\
 & K \arrow[u, "s"] \arrow[r] \arrow[d, "d"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N} \arrow[u, equal] \arrow[d, "\ell"] \\
L : & \mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

Here $\ell(n) = n$ if $n \leq k$ and $\ell(n) = k$ for $n > k$. We have $K = \{(n,i) \mid i < n, i < k\}$, and $d(n,i) = (\ell(n), i)$. The map s is the evident inclusion $(n,i) \mapsto (n,i)$.

Truncate to even length. If the list is of even length, leave it alone; if it is of odd length, delete the last entry. This transformation is given by

$$\begin{tikzcd}
L : & \mathbb{N}' \arrow[r] & \mathbb{N} \\
 & K \arrow[u, "s"] \arrow[r] \arrow[d, "d"'] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N} \arrow[u, equal] \arrow[d, "\ell"] \\
L : & \mathbb{N}' \arrow[r] & \mathbb{N}
\end{tikzcd}$$

where $\ell(2n) = 2n$, $\ell(2n+1) = 2n$, and the set K is best understood in terms of the picture



The map d is easy to describe in terms of ℓ, and the map $s : K \to \mathbb{N}'$ is the evident inclusion. In this example, one might ask where the type constructor `even-length-list` appears — it is not the one given by the middle row. Rather it occurs when image-factorising the cartesian part.

6.1.7 Streams. A *stream* is an infinite sequence of things. As an endofunctor it is represented by the map

$$\mathbb{N} \to 1$$

Note that this is not a finitary functor.

6.1.8 A natural transformation from non-empty lists to streams. Given a non-empty list, we can construct a stream by repeating the list over and over again. So if the list is (a, b, c) the stream will be $(a, b, c, a, b, c, a, b, c, \dots)$. If the list is of length p, then the nth entry in the stream will be n mod p. (Here, by n mod p we mean the smallest natural number congruent to n mod p.) Since L has finite arities whereas S has infinite arity, there can be no cartesian natural transformation between them. So the natural transformation we are describing will be given by a diagram

$$\begin{tikzcd}
L_{>0} : & \mathbb{N}'_{>0} \arrow[r] & \mathbb{N}_{>0} \\
& \mathbb{N} \times \mathbb{N}_{>0} \arrow[u, "c"] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N}_{>0} \arrow[u, equal] \arrow[d] \\
S : & \mathbb{N} \arrow[r] & 1
\end{tikzcd}$$

where c is given by $c(n, p) = (p \, , \, n \bmod p)$.

6.1.9 Repeat to fill. For lists longer than 1024 truncate at this length; for shorter lists, repeat until reaching length 1024.

$$\begin{tikzcd}
L_{>0}: & \mathbb{N}'_{>0} \arrow[r] & \mathbb{N}_{>0} \arrow[d, equal] \\
 & 1024 \times \mathbb{N}_{>0} \arrow[u, "c"] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathbb{N}_{>0} \arrow[d] \\
 & 1024 \arrow[r] & 1
\end{tikzcd}$$

where c is given by $c(n,p) = (p\,,\,n \bmod p)$ for $p \leq 1024$, and $c(n,p) = (p,n)$ for $p < 1024$.

Shapely types

See also Cockett [27], Moggi et al. [85]

6.1.10 Shapely functors. We discuss the relationship between polynomial functors and the shapely functors and shapely types of Jay and Cockett [52], [51]. A *shapely functor* [52] is a pullback-preserving functor $F : \mathscr{E}^m \to \mathscr{E}^n$ equipped with a strength. Since, for a natural number n, the discrete power $\mathscr{E}^n$ is equivalent to the slice $\mathscr{E}/n$, where n now denotes the n-fold sum of 1 in $\mathscr{E}$, it makes sense to compare shapely functors and polynomial functors. Since a polynomial functor preserves pullbacks and has a canonical strength, it is canonically a shapely functor. It is not true that every shapely functor is polynomial. For a counter example, let K be a set with a non-principal filter $\mathbb{D}$, and consider the filter-power functor

$$\begin{array}{rcl} F : \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & \operatorname*{colim}_{D \in \mathbb{D}} X^D \,, \end{array}$$

which preserves finite limits since it is a filtered colimit of representables. Since every endofunctor on ***Set*** has a canonical strength, F is a shapely functor. However, F does not preserve all cofiltered limits, and hence, by 8.6.3 (ii) cannot be polynomial. For example, $\emptyset = \lim_{D \in \mathbb{D}} D$ itself is not preserved. This example is apparently at odds with Theorem 8.3 of [2].

More details on the filter argument. Let $\mathbb{D}$ be a non-principal filter on a set K. For example, let K be an infinite set, and let $\mathbb{D}$ be the filter consisting of all complements of finite subsets of K. Consider the filter-power functor

$$\begin{aligned} F : \textbf{\textit{Set}} &\longrightarrow \textbf{\textit{Set}} \\ X &\longmapsto \operatorname*{colim}_{D\in\mathbb{D}} X^D \end{aligned}$$

[REF: Adamek–Koubek–Trnkova: How large are left exact functors?, TAC 2001.] In the example, $\operatorname{colim}_{D\in\mathbb{D}} X^D$ is the set of almost-everywhere defined maps $K \to X$ modulo the equivalence relation identifying two partial maps if they agree almost everywhere. Since F is a filtered colimit of representables, it preserves finite limits, and we also have $F(\emptyset) = \emptyset$. (In the example, this is clear: since K is infinite, the empty subset does not belong to $\mathbb{D}$). We now check that F does not preserve infinite limits. Specifically it does not preserve the limit $\emptyset = \lim_{D\in\mathbb{D}} D$. To see this, observe that for every $D \in \mathbb{D}$, the set $F(D)$ contains the partial map $K \to D$ given by the identity map on D. Since the restriction of the identity map is always the identity map, this gives us an element in the limit set $\lim_{D\in\mathbb{D}} F(D)$, which is therefore nonempty, hence the limit is not preserved.

6.1.11 Shapely types. Let $L : \mathscr{E} \to \mathscr{E}$ denote the *list endofunctor*, $L(X) = \sum_{n\in\mathbb{N}} X^n$, which is the same as what we usually call the free-monoid monad in Example 1.2.8. A *shapely type* [52] in one variable is a shapely functor equipped with a cartesian strong natural transformation to L. A morphism of shapely types is a natural transformation commuting with the structure map to L. The idea is that the shapely functor represents the template or the shape into which some data can be inserted, while the list holds the actual data; the cartesian natural transformation encodes how the data is to be inserted into the template. As emphasized in [85], the cartesian strong natural transformation is part of the structure of a shapely type. Since any functor with a cartesian natural transformation to L is polynomial by Proposition 2.5.1, it is clear that one-variable shapely types are essentially the same thing as one-variable polynomial endofunctors with a cartesian natural transformation to L, and that there is an equivalence of categories between the category of shapely types and the category $\textbf{\textit{Poly}}^{\mathrm{c}}(1,1)/L$.

According to Jay and Cockett [52], a shapely type in m input variables and n output variables is a shapely functor $\mathscr{E}^m \to \mathscr{E}^n$ equipped with a cartesian strong natural transformation to the functor $L_{m,n} : \mathscr{E}^m \to \mathscr{E}^n$

defined by

$$L_{m,n}(X_i \mid i \in m) = \big(L(\textstyle\sum_{i \in m} X_i) \mid j \in n\big)\,,$$

and they motivate this definition by considerations on how to insert data into templates. With the double-category formalism, we can give a conceptual explanation of the formula: writing $u_m : m \to 1$ and $u_n : n \to 1$ for the maps to the terminal object, the functor $L_{m,n} : \mathscr{E}^m \to \mathscr{E}^n$ is nothing but the composite

$$u_n^* \circ L \circ u_{m!} = (u_m, u_n)^* L\,,$$

the base change of L along (u_m, u_n). Hence we can say uniformly that a shapely type is an object in $\boldsymbol{Poly}^{\mathrm{c}}/L$ with endpoints finite discrete objects.

Aiming at finding a good class of recursive data types closed under fixpoint formation, several researchers came to notions of *positive data types*. They are basically defined as the smallest class of functors containing certain constants, identity functors and closed under sums, products, and fixpoint formation. These are also polynomial. These developments found a good common framework in the theory of containers developed by Abbott and his collaborators [1], [4]. Shapely types as well as strictly positive types, are all captured in the notion of container, which is in fact precisely the notion of polynomial functor. The theory of containers benefited from the work of Moerdijk and Palmgren [81].

6.2 Program semantics

The origins of the use of polynomial functors program semantics go back to the '70s, with the work of the Czech school on automata theory, conditions for existence of free algebras, and existence of minimal realisation of machines [7]. The book [79] was quite influential, and has polynomial functors explicitly. The use of polynomial functors in program semantics was recently explored from a different viewpoint under the name 'interaction systems' in the setting of dependent type theory by Hancock and Setzer [47] and by Hyvernat [50], where polynomials are also given a game-theoretic interpretation. The morphisms there are certain bisimulations, more general than the strong natural transformations used in the present work.

Here is the idea of the game interpretation: for a polynomial

$$\begin{tikzcd} S & D \arrow[l] \arrow[r] & A \arrow[r] & S \end{tikzcd}$$

S is the set of states, A is the set of white moves, D is the set of pairs consisting of a white move followed by a black move. The maps are: from A to S, return the initial state of the white move. From D to A, return the white move. From D to S: return the new state (after the black move). The morphisms are quite different from the usual ones between polynomial functors. They are motivated by the notion of simulation: one game can simulate another if for each state there is a corresponding move giving a bijective set of possible responses... One can think of one game P as being a high-level language and another game Q as a low-level language. Then a simulation from P to Q is an implementation of the high-level language in the low-level language. This game-theoretic use of bridge diagrams is also closely related to something called predicate transformers and to formal topology! Please visit Hyvernat's web page at http://www.lama.univ-savoie.fr/~hyvernat/

6.2.1 Coalgebras. A coalgebra for a polynomial functor $P : \textbf{\textit{Set}} \to \textbf{\textit{Set}}$ is just a set X equipped with a map $X \to P(X)$. There is the notion of terminal object in the category of coalgebras, also called *largest fixpoint*. A dual construction of that iteration may give a way of constructing terminal coalgebras, but I haven't checked under what conditions it works.

We observed for a surjective map $p : E \to B$, the initial algebra is just $\emptyset$. Since $P(\emptyset) \simeq \emptyset$, this is a fixpoint in any case, but as you can imagine it is not the largest one! Such polynomial functors have dull algebras but interesting coalgebras. Example. Consider the identity map $B \to B$. The corresponding polynomial functor is $X \mapsto B \times X$. This particular functor $P(X) = B \times X$ preserves sequential limits and epimorphisms, so in this case we can construct the largest fixpoint in a way dual to the Lambek iteration.

$$\begin{tikzcd} 1 & P(1) \arrow[l] & P(P(1)) \arrow[l] & \ldots \arrow[l] \end{tikzcd}$$

which is just the sequence $1 \longleftarrow B \longleftarrow B^2 \longleftarrow$. Clearly the limit is $B^{\mathbb{N}}$, the set of infinite sequences in B. The isomorphism

$$B^{\mathbb{N}} \simeq B \times B^{\mathbb{N}}$$

is just the exponentiation of the bijection $\mathbb{N} \simeq 1 + \mathbb{N}$.

Algebras and coalgebras for endofunctors $\textbf{\textit{Set}} \to \textbf{\textit{Set}}$ are important concepts in theoretical computer science. Algebras are used as an abstraction of the notion of data types, while coalgebras are an abstraction of systems or automata. Indeed, we saw that an algebra can be interpreted as a set with a collection of operations on it—this is about the same thing as a data type is (e.g., queues and stacks and trees and so on).

6.2.2 Automata. Consider a automaton, like for example a coffee machine. To describe it we first describe its user interface: this consists of an input alphabet I and an output alphabet O. Think of the input alphabet as the set of buttons you can press, and the output alphabet as the set of things that can come out of the machine, i.e. various types of hot drinks, or change. Now let S denote the set of states of the machine, i.e. how much coffee, sugar, and milk the machine contains, as well as water temperature, etc. Now the machine is described by its transition function

$$I \times S \to O \times S.$$

It says that if you press one button while the machine is in a certain state, then something comes out, and the machine enters a new state. By adjunction, the transition function can also be described as

$$S \to (O \times S)^I.$$

In other words, the machine is described as a coalgebra for the polynomial endofunctor $X \mapsto (O \times X)^I$.

Now the theory of polynomial functors says that these exists a terminal coalgebra. This amounts to saying there exists a universal coffee machine! In other words, there exists a coffee machine U (i.e. a set U with a map $U \to (O \times U)^I$) such that for every other machine $S \to (O \times S)^I$ there is a map $S \to U$ commuting with the structure map. So the universal machine can do everything any other machine can do! This machine U will typically be immense, so you might not want to build on for your math department, but even if you don't build one, it is of considerable theoretical interest: namely any other machine S comes with a morphism to U, called the *behaviour* of S. This map does not characterise the machine completely, but the image in U says everything about what you get out of the machince when a given button is pressed. It does not say anything how the machine actually implements this behaviour. Often you are more interested in the behaviour than in the machine itself. . .

Initial algebras are particularly important because they allow for definition and proof by induction. (Similarly terminal coalgebras allow for coinduction. . .) It seems that in practice, algebras modelling data types often come from polynomial functors. For coalgebras modelling systems, it is perhaps less common to be polynomial??

Chapter 7

[Species. . .]

There is scheduled here a long section on species, and comparison between polynomial functors and species.

The following is just copy and paste from [39]:

7.1 Introduction to species and analytical functors

7.1.1 Species. Recall [54], [19] that a *species* is a functor $F : \boldsymbol{FinSet}_{\mathrm{bij}} \to \boldsymbol{Set}$, or equivalently, a sequence $(F[n] \mid n \in \mathbb{N})$ of $\boldsymbol{Set}$-representations of the symmetric groups.

7.1.2 Analytical functors. To a species is associated an *analytic functor*

$$\begin{array}{rcl} \boldsymbol{Set} & \longrightarrow & \boldsymbol{Set} \\ X & \longmapsto & \sum_{n\in\mathbb{N}} F[n] \times_{\mathfrak{S}_n} X^n \,. \end{array}$$

Species and analytic functors were introduced by Joyal [55], who also proved the following theorem characterising analytic functors:

7.1.3 Theorem. *A functor* $\boldsymbol{Set} \to \boldsymbol{Set}$ *is analytic if and only if it preserves weak pullbacks, cofiltered limits, and filtered colimits.*

7.2 Polynomial functors and species

It is the presence of group actions that makes the preservation of pullbacks weak, in contrast to the polynomial functors, cf. (ii) above.

7.2.1 Flat species. Species for which the group actions are free are called *flat* species [19]; they encode rigid combinatorial structures, and correspond to ordinary generating functions rather than exponential ones. The analytic functor associated to a flat species preserves pullbacks strictly and is therefore the same thing as a finitary polynomial functor on ***Set***.

7.2.2 Flat analytic functors and polynomial functors. Explicitly, given a one-variable finitary polynomial functor $P(X) = \sum_{a \in A} X^{B_a}$ represented by $B \to A$, we can 'collect terms': let A_n denote the set of fibres of cardinality n, then there is a bijection

$$\sum_{a \in A} X^{B_a} \cong \sum_{n \in \mathbb{N}} A_n \times X^n .$$

The involved bijections $B_a \cong n$ are not canonical: the degree-n part of P is rather a $\mathfrak{S}_n$-torsor, denoted $P[n]$, and we can write instead

$$P(X) \cong \sum_{n \in \mathbb{N}} P[n] \times_{\mathfrak{S}_n} X^n , \tag{7.1}$$

equ:species

which is the analytic expression of P.

7.2.3 Example: Catalan numbers. As an example of the polynomial encoding of a flat species, consider the species C of binary planar rooted trees. The associated analytic functor is

$$X \mapsto \sum_{n \in \mathbb{N}} C[n] \times_{\mathfrak{S}_n} X^n ,$$

where $C[n]$ is the set of ways to organise an n-element set as the set of nodes of a binary planar rooted tree; $C[n]$ has cardinality $n!\, c_n$, where c_n are the Catalan numbers $1, 1, 2, 5, 14, \ldots$ The polynomial representation is

$$1 \longleftarrow B \longrightarrow A \longrightarrow 1$$

where A is the set of isomorphism classes of binary planar rooted trees, and B is the set of isomorphism classes of binary planar rooted trees with a marked node.

Part II

Polynomial functors in many variables

Chapter 8

Polynomials in many variables

many-variables

8.1 Introductory discussion

We shall now study polynomial in many variables. The idea is that instead of having just one variable X, we have a bigger set of variables $(X_i \mid i \in I)$. For example we might have just two variables X_1 and X_2, corresponding to the indexing set $I = \{1, 2\}$, and then a polynomial functor should be something like

$$\begin{array}{rcl} \textbf{\textit{Set}} \times \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ (X_1, X_2) & \longmapsto & \sum_{b \in B} X_1^{E_{1b}} X_2^{E_{2b}}. \end{array}$$

For each $b \in B$, we specify one exponent set for X_1 and one exponent set for X_2. Hence the whole thing is represented by two maps, $E_1 \to B$ and $E_2 \to B$. We can organise those two maps into a single map $E = E_1 + E_2 \to B$. By construction, the new set E is partitioned into two parts. We encode this by giving a map $s : E \to I$, then the indices on E become an instance of our general notation for fibres: E_1 is the fibre of s over 1, and E_2 is the fibre over 2. The map s now takes care of the bookkeeping of the involved variables: we can write

$$(X_i \mid i \in I) \longmapsto \sum_{b \in B} \prod_{e \in E_b} X_{s(e)}.$$

This expression makes sense for any indexing set I. The data required to represent such an expression is a diagram of shape

$$\begin{tikzcd}
 & E \arrow[ld, "s"'] \arrow[r, "p"] & B \\
I & &
\end{tikzcd}$$

The polynomial takes as input an I-indexed family of sets, and returns a single set.

We will be slightly more general, allowing also for functors returning a J-indexed family of sets. This generality is convenient because it is the natural setting for substituting many-variable polynomials into each other. In particular we will be interested in the case $J = I$, and study polynomial monads in the many-variable setting.

The final definition of a polynomial functor is thus

$$\begin{array}{ccc} \boldsymbol{Set}/I & \longrightarrow & \boldsymbol{Set}/J \\ (X_i \mid i \in I) & \longmapsto & \big(\sum_{b \in B_j} \prod_{e \in E_b} X_{s(e)} \mid j \in J \big), \end{array} \tag{8.1}$$

referring to a diagram of sets and set maps

$$\begin{tikzcd}
 & E \arrow[ld, "s"'] \arrow[r, "p"] & B \arrow[rd, "t"] & \\
I & & & J
\end{tikzcd} \tag{8.2}$$

We shall see in a moment that the big formula is obtained from the diagram by

$$t_! \circ p_* \circ s^*.$$

8.1.1 Slices and presheaf categories. So far we have been talking about I-indexed families of sets, without being precise about what that means. There are two (equivalent) alternatives: one is to encode an I-indexed family $(X_i \mid i \in I)$ as a map $X \to I$. The members of the family are then the fibres. Then the category is just the slice category $\boldsymbol{Set}/I$. The other option is to regard a family as a map $I \to \boldsymbol{Set}$, then the ith member of the family is the image of i. We regard this as a presheaf category: interpreting I as a discrete category, we are talking about the category $\boldsymbol{Fun}(I^{\text{op}}, \boldsymbol{Set})$, i.e. presheaves on I. (The op is irrelevant, but helpful psychologically. We will study non-discrete cases in Part III.)

There is a natural equivalence of categories

$$\textbf{\textit{Set}}^I \simeq \textbf{\textit{Set}}/I.$$

In each case the objects are families of sets $(X_i \mid i \in I)$. In each case a map from $(X_i \mid i \in I)$ to $(Y_i \mid i \in I)$ is a family of maps $(f_i : X_i \to Y_i \mid i \in I)$.

From $\textbf{\textit{Set}}/I$ to $\textbf{\textit{Set}}^I$ there is a canonical functor, associating to a map $X \to I$ the functor $I \to \textbf{\textit{Set}}$ that sends i to X_i (the fibre over i). To construct the inverse functor we need to assemble all the sets X_i into a single set with a map to I. This set should of course be the disjoint union $\coprod_{i \in I} X_i$, but this is only well-defined up to unique isomorphism. . .

REMARK: this adjointness (in this case an equivalence) between taking fibres and taking sums is in fact an instance of, or a phenomenon analogous to, the adjunction $p_! \dashv p^*$. . . Analyse this more carefully. . .

Now some phenomena are easier to grasp in terms of $\textbf{\textit{Set}}^I$ and other are more easily understood in terms of $\textbf{\textit{Set}}/I$. An example of a notion that is best understood in $\textbf{\textit{Set}}^I$ is this: given two I-indexed families, E and X. What is the object X^E? Let's try in $\textbf{\textit{Set}}/I$. So we have families $E \to I$ and $X \to I$. What is the object $X^E \to I$? It is not just the set of maps $E \to X$ over I, because this set does not have a natural map to I, and hence is not an object of $\textbf{\textit{Set}}/I$. In other words, the hom set $\mathrm{Hom}_I(E, X)$ is not internal hom. It is easier to see what it should be in the other viewpoint: the two objects are now maps $X : I \to \textbf{\textit{Set}}$ and $E : I \to \textbf{\textit{Set}}$. The new object X^E should be a functor $I \to \textbf{\textit{Set}}$. Well, $\textbf{\textit{Set}}^I$ is just a presheaf category: the internal hom is pointwise. So it means that for each $i \in I$ we just consider the maps $E(i) \to X(i)$ in $\textbf{\textit{Set}}$. In other words X^E is the functor

$$\begin{array}{ccc} I & \longrightarrow & \textbf{\textit{Set}} \\ i & \longmapsto & X(i)^{E(i)} \end{array}$$

Once we have seen that the exponentiation in $\textbf{\textit{Set}}^I \simeq \textbf{\textit{Set}}/I$ should be fibrewise, we can also describe it in the viewpoint of $\textbf{\textit{Set}}/I$: the power set X^E in $\textbf{\textit{Set}}/I$ is given by specifying that the fibre should be $X_i^{E_i}$. So altogether it is

$$\sum_{i \in I} X_i^{E_i} \to I,$$

so an giving an element amounts to choosing $i \in I$ and then $\varphi : E_i \to X_i$.

(Compare with the 'external' hom set $\mathrm{Hom}_I(E, X) = \prod_{i \in I} X_i^{E_i}$.)

8.1.2 General polynomial functor. Here comes the general construction.

A diagram of sets and set maps like this

$$\begin{tikzcd}
& E \arrow[r, "p"] \arrow[dl, "s"'] & B \arrow[dr, "t"] & \\
I & & & J
\end{tikzcd} \tag{8.3}$$

$\boxed{\texttt{P}}$

gives rise to a *polynomial functor* $P : \textbf{\textit{Set}}/I \to \textbf{\textit{Set}}/J$ defined by

$$\textbf{\textit{Set}}/I \xrightarrow{s^*} \textbf{\textit{Set}}/E \xrightarrow{p_*} \textbf{\textit{Set}}/B \xrightarrow{t_!} \textbf{\textit{Set}}/J.$$

Here lowerstar and lowershriek denote, respectively, the right adjoint and the left adjoint of the pullback functor upperstar. We shall study these adjoints thoroughly in the next section, and extract the explicit formula

$$\begin{array}{ccc} \textbf{\textit{Set}}/I & \longrightarrow & \textbf{\textit{Set}}/J \\ (X_i \mid i \in I) & \longmapsto & \big(\sum_{b \in B_j} \prod_{e \in E_b} X_{s(e)} \mid j \in J \big) \end{array}$$

where $B_j := t^{-1}(j)$, $E_b := p^{-1}(b)$, and $X_i := f^{-1}(i)$.

If I and J are singleton sets, we recover the one-variable polynomial functors of Part 1.

8.2 The pullback functor and its adjoints

8.2.1 Adjunction. Brief reminder—for those of me who is not good at distinguishing left from right. Given two functors

$$\begin{tikzcd}
\mathscr{C} \arrow[r, shift left, "F"] & \mathscr{D} \arrow[l, shift left, "G"]
\end{tikzcd}$$

F is left adjoint to G, written $F \dashv G$, when there is a natural bijection $\mathscr{D}(FX, Y) = \mathscr{C}(X, GY)$. Then for the special case $\mathscr{D}(FX, FX) = \mathscr{C}(X, GFX)$ the element on the right corresponding to id_{FX} is the adjunction's *unit* $1_{\mathscr{C}} \Rightarrow G \circ F$. Similarly there is the *counit* $F \circ G \Rightarrow 1_{\mathscr{D}}$.

$\boxed{\texttt{pullback-fibre}}$

8.2.2 The pullback functor itself. Given $\varphi : A \to B$ and a set Y over B, then

$$\varphi^* Y = A \times_B Y = \sum_{a \in A} Y_{\varphi(a)}.$$

It is often convenient to describe only the fibres: in this case the fibre over $a \in A$ is

$$(\varphi^* Y)_a = Y_{\varphi(a)}.$$

8.2.3 Details on lowershriek. Given $\varphi : A \to B$ and a set $X \to A$, then $\varphi_!(X)$ is described explicitly as

$$\varphi_! X = X = \sum_{b \in B} \sum_{a \in A_b} X_a.$$

In other words, as a set nothing happens; it is only a question about how it is organised into fibres. The fibre over $b \in B$ is

$$(\varphi_! X)_b = \sum_{a \in A_b} X_a.$$

We need to prove that this is really a left adjoint, i.e. establish

$$\mathrm{Hom}_B(\varphi_! X, Y) \ = \ \mathrm{Hom}_A(X, \varphi^* Y).$$

Here is the argument, given

$$\begin{tikzcd}
X \arrow[d] & Y \arrow[d] \\
A \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

then to give $f \in \mathrm{Hom}_B(\varphi_! X, Y)$ is just to provide a commutative square

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

But this is equivalent to giving an A-map from X to the pullback $A \times_B Y$, i.e. a element in the set $\mathrm{Hom}_A(X, \varphi^* Y)$.

t!-pres-pb

8.2.4 Lemma. *The functor $\varphi_! : \textbf{\textit{Set}}/A \to \textbf{\textit{Set}}/B$ preserves pullbacks. More generally it preserves any limit whose shape is a diagram with a terminal object.*

Proof. A pullback in ***Set***$/A$ is just a diagram

$$\begin{tikzcd}
X \times_S Y \arrow[r] \arrow[d] & Y \arrow[d] & \\
X \arrow[r] & S \arrow[r] & A
\end{tikzcd}$$

i.e. a pullback in ***Set*** together with a map from the vertex S to B. When we compose with $A \to B$ it clearly continues to be a pullback square. The same argument works with any limit over a diagram with a terminal object: let the image of that terminal object be S, then the limit is computed in ***Set*** and there is a map from S to A. Applying $\varphi_!$ does not change the limit. □

Note that $\varphi_!$ does not preserve all limits: most blatantly it does not preserve the terminal object of ***Set***$/A$: clearly $\varphi_! A \not\simeq B$ (except if $\varphi \simeq \mathrm{id}$). Products are not preserved either: the products in ***Set***$/A$ are just fibre products over A in ***Set***, and the products in ***Set***$/B$ are fibre products over B. But $\varphi_!(X \times_A Y) \not\simeq X \times_B Y$ (except if $\varphi \simeq \mathrm{id}$).

In fact one can prove more generally that

thm:connectedlimits

8.2.5 Proposition. *The functor $\varphi_!$: **Set**$/A \to$ **Set**$/B$ preserves connected limits.*

This is Carboni–Johnstone [25]. See also Leinster [75]. The proof will soon be included here... Meanwhile:

8.2.6 Exercise. Prove that limits over diagrams of shape

$$\begin{tikzcd}
 & & \cdot \arrow[d] \\
 & \cdot \arrow[r] \arrow[d] & \cdot \\
\cdot \arrow[r] & \cdot &
\end{tikzcd}$$

are reserved by $\varphi_!$. HINT: the limit can be computed as three pullbacks:

first

$$\begin{tikzcd}
& \cdot \arrow[r, dashed] \arrow[d, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] \\
\cdot \arrow[r, dashed] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] & \cdot \\
\cdot \arrow[r] & \cdot &
\end{tikzcd}$$

then discard the two squares (except one side), since we now already have the values at their cone points, and finish by taking a third pullback.

8.2.7 Exercise. Prove that $\varphi_!$ preserves equalisers. HINT: realise it as a pullback.

[If $B = 1$, so that we are talking about $\varphi : A \to 1$, then $\varphi_!$ in fact creates connected limits. Is this true in the general case too? Check it out. . .]

8.2.8 The unit for the adjunction $\varphi_! \dashv \varphi^*$. Start with X over A, lowershriek it to B, and pull it back to A again. By the above formulae, the fibre of $\varphi^*\varphi_! X$ over $a \in A$ is

$$\sum_{q \in A_{\varphi(a)}} X_q$$

The unit for the adjunction is simply the map $X \to \varphi^*\varphi_! X$ defined by sending X_a into the summand corresponding to $q = a$.

8.2.9 The counit for $\varphi_! \dashv \varphi^*$. Starting with Y over B, pull it back to A to get the set φ^*Y, and lowershriek it back to B: this just the same set φ^*Y again, and the counit is the map $\varphi^*Y \to Y$ itself.

We can also give a useful fibrewise description: remembering the formulae $(\varphi_! X)_b = \sum_{a \in A_b} X_a$ and $(\varphi^*Y)_a = Y_{\varphi(a)}$ we find

$$(\varphi_! \varphi^* Y)_b = \sum_{a \in A_b} (\varphi^*Y)_a = \sum_{a \in A_b} Y_{\varphi(a)} = \sum_{a \in A_b} Y_b = A_b \times Y_b.$$

Note that the counit is just the projection $A_b \times Y_b \to Y_b$.

riek-star-cart

8.2.10 Lemma. *The unit and the counit for $\varphi_! \dashv \varphi^*$ are both cartesian natural transformations.*

Proof. Let us first look at the unit: the component of η at an object $p : X \to A$ is the map

$$\begin{array}{rcl} X & \longrightarrow & A \times_B X \\ x & \longmapsto & (p(x), x)) \end{array}$$

so given a map $f : X' \to X$ over A, the corresponding naturality square is

$$\begin{tikzcd} X' \arrow[r] \arrow[d] & A \times_B X' \arrow[d] \\ X \arrow[r] & A \times_B X. \end{tikzcd}$$

We check it is a pullback square by analysing the fibres for the two horizontal maps: starting with (a, x') in $A \times_B X'$, the fibre is singleton if $a = pfx'$ and empty otherwise. In the lower row, the fibre over the image, (a, fx'), is singleton if $a = pfx'$ and empty otherwise. So the fibres are always isomorphic, and we conclude that the square is cartesian.

Consider now the counit: for a map $Y' \to Y$ over B, the naturality square is the top square in the diagram

$$\begin{tikzcd} \varphi^* Y' \arrow[r] \arrow[d] & Y' \arrow[d] \\ \varphi^* Y \arrow[r] \arrow[d] & Y \arrow[d] \\ A \arrow[r, "\varphi"'] & B; \end{tikzcd}$$

The top square is cartesian since the bottom square and the big square are so (cf. A.1). □

8.2.11 Lemma. *If $\varphi : A \to B$ is mono, then the unit* $\mathrm{id}_A \Rightarrow \varphi^* \circ \varphi_!$ *is an isomorphism.*

Proof. □

8.2.12 Details on the lowerstar. Given $\varphi : A \to B$, and a map $X \to A$. Then $\varphi_* X$ is the set over B given by

$$\varphi_* X = \sum_{b \in B} \prod_{a \in A_b} X_a .$$

The slogan for φ_* is: *multiply the fibres*.

To give an element in $\prod_{a \in A_b} X_a$ is to give for each $a \in A_b$ an element in X lying over a itself, so we can also describe the fibre $(\varphi_* X)_b$ as the set of all maps $s : A_b \to X$ making this triangle commute:

$$\begin{tikzcd}
 & X \arrow[d] \\
A_b \arrow[ur, "s"] \arrow[r, phantom, "\subset"] & A
\end{tikzcd}$$

Note that in general there is no canonical map $X \to \varphi_* X$ over f. Indeed, to have such a commutative square would mean that the top map is really a B-map from $\varphi_! X \to \varphi_* X$. (That is how $\varphi_!$ is defined.) Now by the adjunction $\varphi^* \dashv \varphi_*$, to give such a map is the same as giving $\varphi^* \varphi_! X \to X$, and there is no way to get a canonical map like this. There *is* a map in the other direction, namely the unit of the adjunction $\varphi_! \dashv \varphi^* \ldots$

8.2.13 Lemma. *Given $\varphi : A \to B$, the functor*

$$\begin{array}{rcl} \varphi_* : \textbf{\textit{Set}}/A & \longrightarrow & \textbf{\textit{Set}}/B \\ X/A & \longmapsto & \displaystyle\sum_{b \in B} \prod_{a \in A_b} X_a \end{array}$$

is right adjoint to φ^*.

Proof. Given

$$\begin{tikzcd}
X \arrow[d] & Y \arrow[d] \\
A \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

we claim there is a natural bijection

$$\mathrm{Hom}_A(\varphi^* Y, X) \simeq \mathrm{Hom}_B(Y, \varphi_* X).$$

Indeed, to give an A-map $\varphi^*Y \to X$ is to give for each $a \in A$ a map $(\varphi^*Y)_a \to X_a$. This is just to give $Y_{\varphi(a)} \to X_a$. So the left-hand side is

$$\prod_{a\in A} X_a^{Y_{\varphi(a)}}.$$

Now split the product into parts, one for each fibre:

$$= \prod_{b\in B}\prod_{a\in A_b} X_a^{Y_{\varphi(a)}}$$

And then we know $\varphi(a) = b$, getting

$$= \prod_{b\in B}\prod_{a\in A_b} X_a^{Y_b}.$$

On the other hand, to give a B-map $Y \to \varphi_* X = \sum_{b\in B}\prod_{a\in A_b} X_a$, so it is for each $b \in B$ a map $Y_b \to \prod_{a\in A_b} X_a$. A map into a product is the same as a product of maps, so this is the same as the left-hand side. □

8.2.14 Example. Notice how familiar the adjointness becomes when $B = 1$. This is really all there is to it: suppose we have

$$\begin{tikzcd} X \arrow[d] & Y \arrow[d] \\ A \arrow[r, "\varphi"'] & 1. \end{tikzcd}$$

The set $\varphi_* X$ is simply the set of sections to $X \to A$. We denote it $\mathrm{Hom}_A(A, X)$. The adjointness say there is a bijection

$$\mathrm{Hom}_A(A \times Y, X) \leftrightarrow \mathrm{Hom}(Y, \mathrm{Hom}_A(A, X)).$$

But this is easy: given an A-map $\gamma : A \times Y \to X$ we construct

$$\begin{array}{rcl} Y & \longrightarrow & \mathrm{Hom}_A(A, X) \\ y & \longmapsto & [a \mapsto \gamma(a, y)] \end{array}$$

and given for each y an A-map $\sigma_y : A \to X$, we define

$$\begin{array}{rcl} A \times Y & \longrightarrow & X \\ (a, y) & \longmapsto & \sigma_y(a). \end{array}$$

8.2.15 Lemma. *The functor $\varphi_* : \textbf{Set}/A \to \textbf{Set}/B$ preserves filtered colimits if and only if the map $\varphi : A \to B$ has finite fibres.*

Proof. By the formula $\varphi_*(X) = \sum_{b\in B} \prod_{a\in A_b} X_a$, since sums commute with filtered colimits, we see that we can reduce to the case $B = 1$. Now the lemma follows by observing that it is just a product, and products commute with filtered colimits precisely when they are finite.

□

8.2.16 Example. For each infinite set N, it is easy to construct an N-indexed family of directed systems such that the product of all the colimits is not the colimit of the product of the directed systems. For simplicity take $N = \mathbb{N}$. For each $n \in \mathbb{N}$ consider the directed system D_n

$$\cdots = \emptyset = \emptyset \to 1 = 1 \cdots$$

where the first singleton set occurs in position n. It is clear that each of these has colimit 1, so $\prod_{n\in\mathbb{N}} \operatorname{colim}_i D_n = 1$. On the other hand, the product of those directed systems is computed entry-wise, and since for any position i there is a system with $\emptyset$ in position i, the product of all the systems is the system which is constant $\emptyset$, and clearly

$$\operatorname*{colim}_i (\prod_{n\in\mathbb{N}} D_n) \operatorname*{colim}_i \emptyset = \emptyset.$$

8.2.17 The unit for the adjunction $\varphi^* \dashv \varphi_*$. Start with Y over B, pull back to A, to get $\sum_{a\in A} Y_{\varphi(a)}$, then push forth again to get

$$\varphi_*\varphi^*Y = \sum_{b\in B} \prod_{a\in A_b} Y_{\varphi(a)} = \sum_{b\in B} \prod_{a\in A_b} Y_b = \sum_{b\in B} Y_b^{A_b}.$$

Let us also put the fibre in a displayed formula:

$$(\varphi_*\varphi^*Y)_b = Y_b^{A_b}.$$

The unit for the adjunction $\varphi^* \dashv \varphi_*$ is the natural map $Y \to \varphi_*\varphi^*Y$ that sends an element $y \in Y_b$ to the constant map $A_b \to Y_b$ on y.

8.2.18 The counit for $\varphi^* \dashv \varphi_*$ (the evaluation map). Start with X over A; push forth to B and pull back to A again:

$$\varphi^*\varphi_*X = A \times_B \varphi_*X$$

Before being explicit with sums and products, we can already describe the counit $\varphi^*\varphi_*X \to X$: recall that the fibre of φ_*X over b consists of maps $s : A_b \to X$ over A. Now our fibre product over B consists of pairs (a, s) where $a \in A_b$ and $s : A_b \to X$ is in φ_*X. Now the counit for the adjunction is just the evaluation map

$$\begin{aligned} \varepsilon_X : A \times_B \varphi_*X &\longrightarrow X \\ (a, s) &\longmapsto s(a). \end{aligned}$$

And here comes the fibrewise description: the fibre of φ_*X over b is $\prod\limits_{a\in A_b} X_a$. Hence by 8.2.2 the fibre of $\varphi^*\varphi_*X$ over a is

$$\prod_{q\in A_{\varphi(a)}} X_q.$$

The counit $\varepsilon_X : \varphi^*\varphi_*X \to X$ is described fibrewise as the natural projection from $\prod\limits_{q\in A_{\varphi(a)}} X_q$ onto the distinguished factor X_a corresponding to $q = a$.

8.2.19 Locally cartesian closed categories. It is clear that the notion of polynomial functor makes sense anywhere we have a pullback functor having both adjoints. So first of all we need to place ourselves in a category $\mathscr{E}$ with pullbacks. It is not difficult to see that our construction of the left adjoint to pullback is completely formal. Precisely,

8.2.20 Lemma. *If $\mathscr{E}$ is a category with pullbacks, then for any arrow $\varphi : A \to B$ in $\mathscr{E}$ the pullback functor $\varphi^* : \mathscr{E}/B \to \mathscr{E}/A$ has a left adjoint, which is merely*

$$\begin{aligned} \varphi_! : \mathscr{E}/A &\longrightarrow \mathscr{E}/B \\ [X \to A] &\longmapsto [X \to A \to B]. \end{aligned}$$

Hence, so far we need $\mathscr{E}$ to have pullbacks. This is equivalent to saying that every slice of $\mathscr{E}$ has finite products.

What do we need in order to ensure that pullback also has a right adjoint? The construction suggested that this has to do with exponentiation, i.e. that the products existing in each slice are *closed*. That is, in each slice there is a right adjoint to taking product with some fixed object.

A category $\mathscr{E}$ is called *locally cartesian closed* if every slice of it is cartesian closed.

8.2.21 Proposition. (Freyd [37].) *A category $\mathscr{E}$ is locally cartesian closed if and only if it has pullbacks for every arrow $f : X \to Y$ and the pullback functor $f^* : \mathscr{E}/Y \to \mathscr{E}/X$ has a right adjoint.*

Proof. See Barr–Wells [14], Theorem 13.4.3. □

Our favourite category ***Set*** is of course locally cartesian closed. More generally, any topos is locally cartesian closed. In contrast, ***Cat*** is cartesian closed but not locally cartesian closed, cf. Example 15.0.2. Although we are mainly interested in the case $\mathscr{E} = \textbf{\textit{Set}}$, it is very useful to work out as much of the theory as possible in the general setting. It allows us to see clearly which results are completely formal, and which depend on particular features of ***Set***.

On the other hand, we also use many arguments involving formulae like

$$\left(\sum_{b \in B_j} \prod_{e \in E_b} X_{s(e)} \mid j \in J \right)$$

which look completely set theoretic! But in fact such formulae can be interpreted in any locally cartesian closed category $\mathscr{E}$, by exploiting the so-called internal language of $\mathscr{E}$.

Currently these notes do not go into these issues, since we are content with developing the theory only in the setting of ***Set***.

If the category has some sums. . .

Such a category is said to have *dependent products* if the pullback functor has also a right adjoint f_*. A pretopos $\mathscr{E}$ has dependent products if and only if it is locally cartesian closed.

DO SOMETHING WITH THIS! In topos theory, the functor p_* is often denoted Π_p, because it is 'multiply', and sometimes (inspired by the role these maps play in logic) it is also denoted $\forall_p$. The functor $t_!$ is often denoted Σ_t, since it is the sum map, and sometimes it is also denoted $\exists_t$. In logic, these notions make sense more generally in any Heyting pretopos. . . cf. [81]. Finally let us denote the pullback functor Δ_p. It is also called substitution, since it is about substituting one set of variables into another. . . However, this terminology will not be used here, because we are going to talk about substitution of one polynomial into another. . .

, or, which is about the same: give a concise description of the polynomial functor

$$P(X) = \sum_{b \in B} X^{E_b}$$

can be described as the total space of the following object in ***Set***$/B$, namely

$$(X \times B \to B)^{p:E \to B}$$

We have just described this object: its fibre over $b \in B$ is the power set X^{E_b}. That's all.

Repeat this description: given $1 \xleftarrow{s} E \xrightarrow{p} B$, and a set X, we claim that $P(X)$ can be described as an object of ***Set***$/B$ as

$$(s^*X)^E$$

Where $s^*X = X \times E$. We just argued that exponentiation in ***Set***$/B$ is fibrewise. So this object is the set over B whose b-fibre is

$$(s^*X)_b^{E_b}$$

but $(s^*X)_b$ is just X, so the fibre is X^{E_b} and the total space is $\sum_{b \in B} X^{E_b}$.

Coherence

Let f and g be composable arrows in $\mathscr{E}$. Since composition of arrows is strictly associative, it follows that

$$g_! \circ f_! = (g \circ f)_! \, .$$

This is possibly an irrelevant feature. The natural feature is just that these two functors are naturally isomorphic. (For example, if we accept the standpoint that the crucial property of lowershriek is to be left adjoint to upperstar, then if we took another left adjoint (of course necessarily isomorphic to the standard lowershriek), then we would only find isomorphism, not equality.

In any case, for pullback itself, we only have a natural isomorphism

$$f^* \circ g^* \simeq (g \circ f)^*$$

Formally, the association

$$\begin{array}{rcl} \mathscr{E}^{\mathrm{op}} & \longrightarrow & \textbf{\textit{Cat}} \\ A & \longmapsto & \mathscr{E}/A \\ f & \longmapsto & f^* \end{array}$$

is a pseudo-functor, not a strict functor. It means that those natural isomorphisms are built in as a part of the structure.

From those canonical and invertible 2-cells, we get other 2-cells. For example, for every commutative square

$$\begin{tikzcd}
\cdot \arrow[r, "\overline{\varphi}"] \arrow[d, "\alpha"'] & \cdot \arrow[d, "\beta"] \\
\cdot \arrow[r, "\varphi"'] & \cdot
\end{tikzcd}$$

we get a natural isomorphism

$$\overline{\varphi}^* \circ \beta^* \simeq \alpha^* \circ \varphi^*$$

8.3 Beck-Chevalley and distributivity

In this section we should work in a general locally cartesian closed category $\mathscr{E}$.

BC **8.3.1 The Beck-Chevalley condition.** Given a pullback square

$$\begin{tikzcd}
\overline{A} \arrow[r, "\overline{\varphi}"] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & \overline{B} \arrow[d, "\beta"] \\
A \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

there are natural isomorphisms of functors

$$\alpha_! \circ \overline{\varphi}^* \stackrel{\sim}{\to} \varphi^* \circ \beta_! \qquad \text{and} \qquad \beta^* \circ \varphi_* \stackrel{\sim}{\to} \overline{\varphi}_* \circ \alpha^*.$$

They are usually referred to as the Beck-Chevalley conditions.

Here is an elementary proof of the first. (The two isomorphisms determine each other via adjunction.) Let $Y \to \overline{B}$ be an object in $\mathscr{E}/\overline{B}$. Consider the diagram

$$\begin{tikzcd}
\overline{A} \times_{\overline{B}} Y \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & Y \arrow[d] \\
\overline{A} \arrow[r] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & \overline{B} \arrow[d, "\beta"] \\
A \arrow[r] & B
\end{tikzcd}$$

Applying $\alpha_! \circ \overline{\varphi}^*$ to $Y \to \overline{B}$ amounts to first performing the pullback of the top square, then composing the left-hand side with α. The result is the long left-hand side of the diagram. On the other hand, applying $\varphi^* \circ \beta_!$ to $Y \to \overline{B}$ amounts to first composing with β, then performing the pullback of the outer square. The equality of these two constructions is nothing but the fact that the pasting of two pullback squares is again a pullback square.

We shall analyse the Beck-Chevalley situation further, and see that the statement follows formally just from the adjunctions. Consider now an arbitrary (commutative) square

$$\begin{tikzcd}
\overline{A} \arrow[r, "\overline{\varphi}"] \arrow[d, "\alpha"'] & \overline{B} \arrow[d, "\beta"] \\
A \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

We have tautologically a commuting diagram of functors

$$\begin{tikzcd}
\mathscr{E}/\overline{A} \arrow[r, "\overline{\varphi}_!"] \arrow[d, "\alpha_!"'] & \mathscr{E}/\overline{B} \arrow[d, "\beta_!"] \\
\mathscr{E}/A \arrow[r, "\varphi_!"'] \arrow[ur, phantom, "\Nearrow"] & \mathscr{E}/B
\end{tikzcd}$$

Taking mate (with respect to the adjunctions $\overline{\varphi}_! \dashv \overline{\varphi}^*$ and $\varphi_! \dashv \varphi^*$) we get a 2-cell

$$\begin{tikzcd}
\mathscr{E}/\overline{A} \arrow[d, "\alpha_!"'] \arrow[dr, phantom, "\Searrow"] & \mathscr{E}/\overline{B} \arrow[l, "\overline{\varphi}^*"'] \arrow[d, "\beta_!"] \\
\mathscr{E}/A & \mathscr{E}/B \arrow[l, "\varphi^*"]
\end{tikzcd}$$

This is the natural transformation of the Beck-Chevalley condition, so we have established that it always exists in the direction indicated. We have also shown that it is cartesian (since the adjunction used to take mate has cartesian unit and counit). In fact, the mate business amounts to saying that the Beck-Chevalley 2-cells is

$$\alpha_! \circ \overline{\varphi}^* \overset{\text{unit}}{\Rightarrow} \varphi_* \circ \varphi_! \circ \alpha_! \circ \overline{\varphi}^* = \varphi_* \circ \beta_! \circ \overline{\varphi}_! \circ \overline{\varphi}^* \overset{\text{counit}}{\Rightarrow} \varphi^* \circ \beta_!$$

(The remaining statement is that this natural transformation is invertible when the original square is a pullback. We already proved that by a direct argument.)

Now we can take mate another time, this time with respect to $\alpha_! \dashv \alpha^*$ and $\beta_! \dashv \beta^*$. We get

$$\begin{tikzcd}
\mathscr{E}/\overline{A} & \mathscr{E}/\overline{B} \arrow[l, "\overline{\varphi}^*"'] \\
\mathscr{E}/A \arrow[u, "\alpha^*"] \arrow[ur, phantom, "\Swarrow"] & \mathscr{E}/B \arrow[l, "\varphi^*"] \arrow[u, "\beta^*"']
\end{tikzcd}$$

(This is the obvious isomorphism obtained by pulling back along in two ways along the edges of a commutative square.) And finally take mate with respect to the adjunction $\overline{\varphi}^* \dashv \overline{\varphi}_*$ and $\varphi^* \dashv \varphi_*$ to get

$$\begin{tikzcd}
\mathscr{E}/\overline{A} \arrow[r, "\overline{\varphi}_*"] & \mathscr{E}/\overline{B} \\
\mathscr{E}/A \arrow[u, "\alpha^*"] \arrow[r, "\varphi_*"'] \arrow[ur, phantom, "\Nwarrow"] & \mathscr{E}/B. \arrow[u, "\beta^*"']
\end{tikzcd}$$

This is the natural transformation of the second Beck-Chevalley condition:

$$\beta^* \circ \varphi_* \overset{\text{unit}}{\Rightarrow} \overline{\varphi}_* \circ \overline{\varphi}^* \circ \beta^* \circ \varphi_* \;=\; \overline{\varphi}_* \circ \alpha^* \circ \varphi^* \circ \varphi_* \overset{\text{counit}}{\Rightarrow} \overline{\varphi}_* \circ \alpha^*$$

It always exists as a 2-cell in the indicated direction (but this one is not in general cartesian!), and it remains to establish that it is invertible when the original square is a pullback.

`lowershriek-lowerstar`

8.3.2 Lemma. *Given a pullback square*

$$\begin{tikzcd}
\overline{A} \arrow[r, "\overline{\varphi}"] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & \overline{B} \arrow[d, "\beta"] \\
A \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

there is induced a CARTESIAN natural transformation

$$\beta_! \circ \overline{\varphi}_* \Rightarrow \varphi_* \circ \alpha_! \,.$$

Proof. There are two ways to get it, and both depend on the inverse of the Beck-Chevalley transformations. Start with the inverse 2-cell of the Beck-Chevalley transformation corresponding to the left adjoints:

$$\begin{tikzcd}
\mathscr{E}/\overline{A} \arrow[d, "\alpha_!"'] \arrow[dr, phantom, "\Nwarrow"] & \mathscr{E}/\overline{B} \arrow[l, "\overline{\varphi}^*"'] \arrow[d, "\beta_!"] \\
\mathscr{E}/A & \mathscr{E}/B \arrow[l, "\varphi^*"]
\end{tikzcd}$$

Now take mate with respect to $\overline{\varphi}^* \dashv \overline{\varphi}_*$ and $\varphi^* \dashv \varphi_*$, getting the promised 2-cell:

$$\begin{tikzcd}
\mathscr{E}/\overline{A} \arrow[r, "\overline{\varphi}_*"] \arrow[d, "\alpha_!"'] \arrow[dr, phantom, "\Swarrow"] & \mathscr{E}/\overline{B} \arrow[d, "\beta_!"] \\
\mathscr{E}/A \arrow[r, "\varphi_*"'] & \mathscr{E}/B
\end{tikzcd}$$

We could also arrive at this 2-cell by starting with the inverse of the Beck-Chevalley transformation corresponding to the right adjoints:

$$\begin{tikzcd}
\mathscr{E}/\overline{A} \arrow[r, "\overline{\varphi}_*"] & \mathscr{E}/\overline{B} \\
\mathscr{E}/A \arrow[u, "\alpha^*"] \arrow[r, "\varphi_*"'] \arrow[ur, phantom, "\Searrow"] & \mathscr{E}/B. \arrow[u, "\beta^*"']
\end{tikzcd}$$

and then taking mate with respect to $\alpha_! \dashv \alpha^*$ and $\beta_! \dashv \beta^*$. This second construction shows that it is cartesian, because it is made up of cartesian cells.

(One can wonder whether these two constructions actually agree, or if there are in fact *two* natural transformations $\beta_! \circ \overline{\varphi}_* \Rightarrow \varphi_* \circ \alpha_!$. The answer is that they do agree. This is subtle to prove: it depends on a compatibility between inverses and mates, and in general taking mates does not preserve invertibility... Perhaps this proof should go into the appendix.) □

8.3.3 Logical and graphical interpretation of this lemma. Let us take the particular case where $B = 1$. This is already enough to see the essentials, since it is really a fibre-wise issue. So given a square

$$\begin{tikzcd}
S \times T \arrow[r, "q"] \arrow[d, "p"'] \arrow[dr, phantom, "\lrcorner", very near start] & T \arrow[d, "v"] \\
S \arrow[r, "u"'] & 1
\end{tikzcd}$$

The claim is that the natural transformation

$$v_! \circ q_* \Rightarrow u_* \circ p_!$$

is cartesian. We write it out in the internal language: for $X \to S \times T$, the map is

$$\sum_{t \in T} \prod_{s \in S} X_{(s,t)} \longrightarrow \prod_{s \in S} \sum_{t \in T} X_{(s,t)}$$

The left-hand side is: pick one $t \in T$, and then for each $s \in S$ an element in $X_{(s,t)}$. That's an element in each of the entries in the t row of X. The right-hand side is: for each $s \in S$, pick an element in some $X_{(s,t_s)}$, where that t_s can be chosen freely for each s. In other words, we need to choose an element in each column of X. The left-hand side choice is of course a special case of the right-hand side choice, namely the special case where all the t_s are the same, so that the elements all belong to the same row.

Now we understand the map, now it is easy to see that it is cartesian, because the component corresponding to a map $X \to Y$ is only about using different sets, the layout is the same for both maps. In other words, all the info is in the shape.

distributivity

8.3.4 Distributivity. Starting from maps $A \xrightarrow{\varphi} B \xrightarrow{\psi} C$, we can construct the following diagram by applying ψ_* to the map $\varphi : A \to B$:

$$\begin{tikzcd}
\psi^*\psi_*A \arrow[r, "\overline{\psi}"] \arrow[d, "\varepsilon_A"'] \arrow[dr, phantom, "\lrcorner", very near start] & \psi_*A \arrow[dd, "\widetilde{\varphi}"] \\
A \arrow[d, "\varphi"'] & \\
B \arrow[r, "\psi"'] & C.
\end{tikzcd}$$

Here ε_A is the A-component of the counit for the adjunction $\psi^* \dashv \psi_*$.

thm:distributive

8.3.5 Proposition. *In the situation just described, the* distributive law *holds for any $X \to A$:*

$$\boxed{\psi_*\varphi_!\,X = \widetilde{\varphi}_!\,\overline{\psi}_*\varepsilon^*X}$$

Proof. From the cartesian square and Beck-Chevalley we get an invertible (and hence cartesian) natural transformation

$$\overline{\psi}_* \circ \varepsilon_A^* \circ \varphi^* \Rightarrow \widetilde{\varphi}^* \circ \psi_*$$

Now take mates with respect to the cartesian adjunctions $\varphi_! \dashv \varphi^*$ and $\widetilde{\varphi}_! \dashv \widetilde{\varphi}^*$ to arrive at the cartesian natural transformation

$$\widetilde{\varphi}_! \circ \overline{\psi}_* \circ \varepsilon^* \Rightarrow \psi_* \circ \varphi_!\,.$$

It remains to show that this is invertible. Since it is cartesian, it is enough to show that its terminal component is invertible. But if we apply $\widetilde{\varphi}_! \circ \overline{\psi}_* \circ \varepsilon_A^*$ to id_A the result is clearly $\widetilde{\varphi}$, which is also what we get from applying $\psi_* \circ \varphi_!$ to id_A. □

8.3.6 Distributivity in the internal language. Distributivity as we know it from elementary school describes how to transform a product of sums into a sum of products. The abstract distributivity formula here does just that, as we see if we write out explicitly what it means. Temporarily, put $Y = \varphi_! X$. The fibre of Y over $b \in B$ is $\sum_{a \in A_b} X_a$. Pushing forth Y to C gives a set with fibre $\prod\limits_{b \in B_c} Y_b$, so in this case the fibre is

$$\prod_{b \in B_c} \sum_{a \in A_b} X_a.$$

Hence the left-hand side of the distributivity statement is

$$\psi_* \varphi_! X = \sum_{c \in C} \prod_{b \in B_c} \sum_{a \in A_b} X_a. \tag{8.4}$$

$\boxed{\texttt{prod-of-sums}}$

Writing only the fibre over $c \in C$, the formula is $\big(\psi_* \varphi_! X\big)_c = \prod_{b \in B_c} \sum_{a \in A_b} X_a$.

Now for the other way around the diagram. We will work through the formula a little backwards, starting with an arbitrary map $Z \to \psi_* A$. Then $\widetilde{\varphi}_! Z$ is the set over C with fibre

$$\sum_{s \in (\psi_* A)_c} Z_s$$

We are interested in the case where $Z = \overline{\psi}_* W$ for some W over $\psi^* \psi_* A$. So in that case the fibre of $\overline{\psi}_* W$ over $s \in \psi_* A$ is

$$\prod_{m \in (\psi^* \psi_* A)_s} W_m$$

Substituting this formula into the previous, we find this expression for the fibre over c of $\widetilde{\varphi}_! \overline{\psi}_* W$:

$$(\widetilde{\varphi}_! \overline{\psi}_* W)_c = \sum_{s \in (\psi_* A)_c} \prod_{m \in (\psi^* \psi_* A)_s} W_m.$$

Here we can simplify the indices in the product a little bit: to give an element in the pullback $\psi^*\psi_* A = B \times_C \psi_* A$ that maps to $s \in \psi_* A$, and knowing that this s maps to c, all we need to specify is the first factor in the fibre product, which of course has to lie in the fibre B_c. So we can write a little more concisely:

$$(\widetilde{\varphi}_! \overline{\psi}_* W)_c = \sum_{s \in (\psi_* A)_c} \prod_{b \in B_c} W_m.$$

Finally, the W we are interested in is $\varepsilon^* X$, whose fibre over $m \in \psi^*\psi_* A$ is $X_{\varepsilon(m)}$. Now the counit ε is just the evaluation map $(b, s) \mapsto s(b)$, so we can write $W_m = X_{\varepsilon(m)} = X_{s(b)}$. So here is the final expansion of the right-hand side of the distributivity statement:

$$(\widetilde{\varphi}_! \overline{\varphi}_* \varepsilon^* X)_c = \sum_{s \in (\psi_* A)_c} \prod_{b \in B_c} X_{s(b)}. \tag{8.5}$$

sum-of-prods

All the above were just preliminary manoeuvres to make explicit what distributivity means. At the same time it made it clear that everything is fibrewise. So in order to prove that (8.4) is equal to (8.5), we can reduce to the case where $C = 1$. In this case there is a simpler description of the lowerstar: $\psi_* A$ is simply the set of sections to $\varphi : A \to B$, which we denote $\Gamma_B(A)$.

Now the statement is all about these two maps:

$$X \xrightarrow{\gamma} A \xrightarrow{\varphi} B$$

and the pushforth to C. Consider it like this:

$$\begin{tikzcd}
X \arrow[rr, "\gamma"] \arrow[dr] & & A \arrow[dl] & & \psi_* X \arrow[rr, "\psi_*(\gamma)"] \arrow[dr] & & \psi_* A \arrow[dl] \\
& B \arrow[rrrr, "\psi"'] & & & & C &
\end{tikzcd}$$

The map $\psi_*(\gamma)$ is just

$$\begin{array}{ccc} \Gamma_B(X) & \longrightarrow & \Gamma_B(A) \\ \sigma & \longmapsto & \gamma \circ \sigma. \end{array}$$

The left-hand side of the distributivity equation is

$$\psi_* \varphi_! X = \prod_{b \in B} \sum_{a \in A_b} X_a = \Gamma_B(X).$$

We want to describe it in terms of pulling back X along ε and then multiplying. In other words, we want to write it as a sum. The sum should be over all $s \in \prod_{b \in B} A_b = \Gamma_B(A)$. Well, there is a natural map $\Gamma_B(X) \to \Gamma_B(A)$ namely $\sigma \mapsto \gamma \circ \sigma$. (Check that this map is $\psi_*(\gamma)$.) So it remains to describe the fibres of this map. Here is the relevant diagram:

$$\begin{tikzcd}
s^*(X) \arrow[r] \arrow[d] & X \arrow[d, "\gamma"] \\
B \arrow[r, "s"'] & A
\end{tikzcd}$$

To give a section $B \to X$ (over s) is the same as giving $B \to s^* X$. So the fibre is

$$\Gamma_B(s^* X) = \prod_{b \in B} X_{s(b)}$$

In total, we have written

$$\Gamma_B(X) = \sum_{s \in \Gamma_B(A)} \prod_{b \in B} X_{s(b)}$$

This is precisely (8.5), the right-hand side of the distributivity equation.

The proof above already implicitly shows how to reduce the expression of distributivity to an expression involving only the two original maps

$$A \xrightarrow{\varphi} B \xrightarrow{\psi} C$$

Here are some more details. Let's put a few more names in the diagram, writing $M = \psi_* A$ and $N = \psi^* M$:

$$\begin{tikzcd}
N \arrow[r, "\overline{\psi}"] \arrow[d, "\varepsilon_A"'] \arrow[dr, phantom, very near start, "\lrcorner"] & M \arrow[dd, "\widetilde{\varphi}"] \\
A \arrow[d, "\varphi"'] & \\
B \arrow[r, "\psi"'] & C.
\end{tikzcd}$$

Spelled out in the 'internal language', distributivity says:

$$\Big(\prod_{b \in B_c} \sum_{a \in A_b} X_a \mid c \in C \Big) = \Big(\sum_{m \in M_c} \prod_{n \in N_m} X_{\varepsilon(n)} \mid c \in C \Big)$$

The right-hand side involves the new sets M and N, but we will now expand these by their definition to arrive at an expression involving only the original sets and maps. By construction, we have

$$M = \Big(\prod_{b \in B_c} A_b \mid c \in C \Big)$$

hence

$$M_c = \{ m : B_c \to A \mid \text{ sections to } \varphi \}.$$

Now to give $n \in N_m$ is the same as giving $b \in B_c$ for $c = \widetilde{\varphi}(m)$. The map $\varepsilon : N \to A$ is defined as evaluation: it sends $n = (m, b)$ to $m(b)$. Hence we can write $\prod_{n \in N_m} X_{\varepsilon(n)} = \prod_{b \in B_c} X_{m(b)}$. Finally we can write the right-hand side of distributivity as

$$= \Big(\sum_{m : B_c \to A} \prod_{b \in B_c} X_{m(b)} \mid c \in C \Big).$$

8.4 Further Beck-Chevalley gymnastics

Sec:gymnastics

The twelve ways of a square

For any square

$$\begin{tikzcd}
\cdot \arrow[r, "f"] \arrow[d, "u"'] & \cdot \arrow[d, "v"] \\
\cdot \arrow[r, "g"'] & \cdot
\end{tikzcd}$$

we denote the Beck-Chevalley transformations

$$\phi : f^* u_! \Rightarrow v_! g^* \qquad \psi : g_* v^* \Rightarrow u^* f_*$$

NOTE THAT WE WRITE COMPOSITION FROM LEFT TO RIGHT!

If the square is a pullback square, then we also have the transformation of Lemma 8.3.2, which we denote by

$$\gamma : f_* v_! \Rightarrow u_! g_* .$$

twelveways

8.4.1 Proposition. *For any square*

$$\begin{tikzcd}
\cdot \arrow[r, "f"] \arrow[d, "u"'] \arrow[dr, phantom, "(\star)"] & \cdot \arrow[d, "v"] \\
\cdot \arrow[r, "g"'] & \cdot
\end{tikzcd}$$

the following eight equations of 2*-cells hold:*

$$\begin{tikzcd}
f^* \arrow[r, Rightarrow, "\eta_v"] \arrow[d, Rightarrow, "\eta_u"'] \arrow[dr, phantom, "\text{(i)}"] & v_! v^* f^* \arrow[d, equal] \\
f^* u_! u^* \arrow[r, Rightarrow, "\phi"'] & v_! g^* u^*
\end{tikzcd}
\qquad
\begin{tikzcd}
v_* g^* u^* \arrow[r, Rightarrow, "\psi"] \arrow[d, equal] \arrow[dr, phantom, "\text{(ii)}"] & f^* u_* u^* \arrow[d, Rightarrow, "\varepsilon_u"] \\
v_* v^* f^* \arrow[r, Rightarrow, "\varepsilon_v"'] & f^*
\end{tikzcd}$$

$$\begin{tikzcd} v^*f^*u_! \arrow[r, Rightarrow, "\phi"] \arrow[d, equal] \arrow[dr, phantom, "\text{(iii)}"] & v^*v_!g^* \arrow[d, Rightarrow, "\varepsilon_v"] \\ g^*u^*u_! \arrow[r, Rightarrow, "\varepsilon_u"'] & g^* \end{tikzcd}$$

$$\begin{tikzcd} g^* \arrow[r, Rightarrow, "\eta_u"] \arrow[d, Rightarrow, "\eta_v"'] \arrow[dr, phantom, "\text{(iv)}"] & g^*u^*u_* \arrow[d, equal] \\ v^*v_*g^* \arrow[r, Rightarrow, "\psi"'] & v^*f^*u_* \end{tikzcd}$$

$$\begin{tikzcd} f_! \arrow[r, Rightarrow, "\eta_v"] \arrow[d, Rightarrow, "\eta_u"'] \arrow[dr, phantom, "\text{(v)}"] & f_!v_!v^* \arrow[d, equal] \\ u_!u^*f_! \arrow[r, Rightarrow, "\phi"'] & u_!g_!v^* \end{tikzcd}$$

$$\begin{tikzcd} u_*g_*v^* \arrow[r, Rightarrow, "\psi"] \arrow[d, equal] \arrow[dr, phantom, "\text{(vi)}"] & u_*u^*f_* \arrow[d, Rightarrow, "\varepsilon_u"] \\ f_*v_*v^* \arrow[r, Rightarrow, "\varepsilon_v"'] & f_* \end{tikzcd}$$

$$\begin{tikzcd} u^*f_!v_! \arrow[r, Rightarrow, "\phi"] \arrow[d, equal] \arrow[dr, phantom, "\text{(vii)}"] & g_!v^*v_! \arrow[d, Rightarrow, "\varepsilon_v"] \\ u^*u_!g_! \arrow[r, Rightarrow, "\varepsilon_u"'] & g_! \end{tikzcd}$$

$$\begin{tikzcd} g_* \arrow[r, Rightarrow, "\eta_u"] \arrow[d, Rightarrow, "\eta_v"'] \arrow[dr, phantom, "\text{(viii)}"] & u^*u_*g_* \arrow[d, equal] \\ g_*v^*v_* \arrow[r, Rightarrow, "\psi"'] & u^*f_*v_* \end{tikzcd}$$

If furthermore the square $(\star)$ *is cartesian, we also have these four equations:*

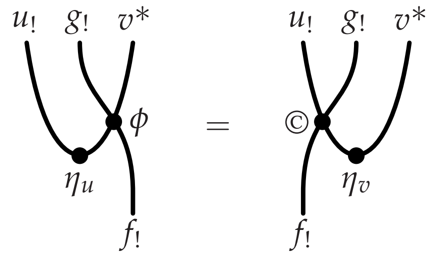
$$\begin{tikzcd} u_*g_!v^* \arrow[r, Rightarrow, "\gamma"] \arrow[d, Rightarrow, "\phi^{-1}"'] \arrow[dr, phantom, "\text{(ix)}"] & f_!v_*v^* \arrow[d, Rightarrow, "\varepsilon_v"] \\ u_*u^*f_! \arrow[r, Rightarrow, "\varepsilon_u"'] & f_! \end{tikzcd}$$

$$\begin{tikzcd} f_* \arrow[r, Rightarrow, "\eta_u"] \arrow[d, Rightarrow, "\eta_v"'] \arrow[dr, phantom, "\text{(x)}"] & u_!u^*f_* \arrow[d, Rightarrow, "\psi^{-1}"] \\ f_*v_!v^* \arrow[r, Rightarrow, "\gamma"'] & u_!g_*v^* \end{tikzcd}$$

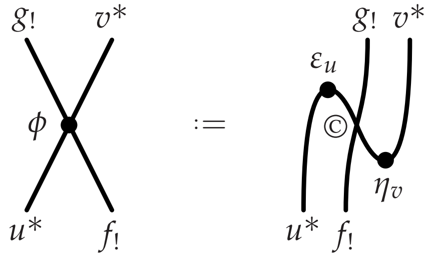
$$\begin{tikzcd} g_! \arrow[r, Rightarrow, "\eta_v"] \arrow[d, Rightarrow, "\eta_u"'] \arrow[dr, phantom, "\text{(xi)}"] & g_!v^*v_* \arrow[d, Rightarrow, "\phi^{-1}"] \\ u^*u_*g_! \arrow[r, Rightarrow, "\gamma"'] & u^*f_!v_* \end{tikzcd}$$

$$\begin{tikzcd} v^*f_*u_! \arrow[r, Rightarrow, "\gamma"] \arrow[d, Rightarrow, "\psi^{-1}"'] \arrow[dr, phantom, "\text{(xii)}"] & v^*v_!g_* \arrow[d, Rightarrow, "\varepsilon_v"] \\ g_*u^*u_! \arrow[r, Rightarrow, "\varepsilon_u"'] & g_* \end{tikzcd}$$

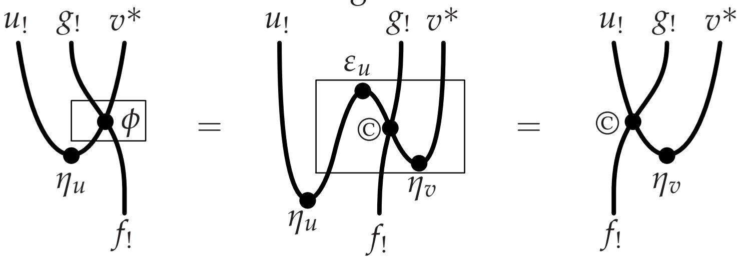
Proof. All the proofs are elementary. As an example let us prove (v). We use string diagrams, read from bottom to top. The claim is



where © denotes the identity 2-cell. The proof is just to use the definition of the 2-cell ϕ:



which is seen here inside the rectangle:



The second step was the triangle identity for the adjunction $u_! \dashv u^*$. □

The six ways of a pair of squares

sixways **8.4.2 Lemma.** *For any pair of squares*

$$\begin{tikzcd} \cdot \arrow[r, "f"] \arrow[d, "u"'] & \cdot \arrow[r, "\tilde{f}"] \arrow[d, "v"] & \cdot \arrow[d, "w"] \\ \cdot \arrow[r, "g"'] & \cdot \arrow[r, "\tilde{g}"'] & \cdot \end{tikzcd}$$

the following four equations of 2*-cells hold:*

$$\begin{tikzcd} u^* f_! \tilde{f}_! \arrow[rr, Rightarrow] \arrow[dr, Rightarrow] & & g_! \tilde{g}_! w^* \\ & g_! v^* \tilde{g}_! \arrow[ur, Rightarrow] & \end{tikzcd} \qquad \begin{tikzcd} w_* \tilde{g}^* g^* \arrow[rr, Rightarrow] \arrow[dr, Rightarrow] & & \tilde{f}^* f^* u_* \\ & \tilde{f}^* u_* g^* \arrow[ur, Rightarrow] & \end{tikzcd}$$

$$\begin{tikzcd}
\tilde{f}^* f^* u_! \arrow[rr, Rightarrow] \arrow[dr, Rightarrow] & & w_! \tilde{g}^* g^* \\
& \tilde{f} v_! g^* \arrow[ur, Rightarrow] &
\end{tikzcd}
\qquad
\begin{tikzcd}
g_* \tilde{g}_* w^* \arrow[rr, Rightarrow] \arrow[dr, Rightarrow] & & u^* f_* \tilde{f}_* \\
& g_* v^* \tilde{f}_* \arrow[ur, Rightarrow] &
\end{tikzcd}$$

Here, the horizontal natural transformations are Beck-Chevalley along a composite map. So the equations just say that a Beck-Chevalley along a composite map can be realised in two steps.

Furthermore, if the squares are pullbacks, then we also have the following two equations:

$$\begin{tikzcd}
f_* \tilde{f}_* w_! \arrow[rr, Rightarrow] \arrow[dr, Rightarrow] & & u_! g_* \tilde{g}_* \\
& f_* v_! g_* \arrow[ur, Rightarrow] &
\end{tikzcd}
\qquad
\begin{tikzcd}
u_* g_! \tilde{g}_! \arrow[rr, Rightarrow] \arrow[dr, Rightarrow] & & f_! \tilde{f}_! w_* \\
& f_! v_* \tilde{g}_! \arrow[ur, Rightarrow] &
\end{tikzcd}$$

One more lemma

Here is a concrete problem to get us started: Suppose we have

$$\begin{tikzcd}
\cdot \arrow[r, "\varphi"] & \cdot \arrow[r, "\psi"] & \cdot \arrow[r, "\psi'"] & \cdot
\end{tikzcd}$$

Then we have an expression

$$\psi'_* \circ \psi_* \circ \varphi_!$$

Now we can rewrite in a single step using distributivity of $(\psi' \circ \psi)_*$ over $\varphi_!$. Or we could use distributivity in two steps, and then a Beck-Chevalley, as follows: consider the diagram (assumed to be the sort of diagram occurring in distributivity):

$$\begin{tikzcd}
\cdot \arrow[r, "s"] \arrow[d, "a"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r, "m"] \arrow[d, "e"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[ddd, "w"] \\
\cdot \arrow[r, "g"] \arrow[d, "\varepsilon"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[dd, "c"] & \\
B \arrow[d, "b"'] & & \\
\cdot \arrow[r, "q"'] & \cdot \arrow[r, "r"'] & \cdot
\end{tikzcd}$$

The upper right-hand corner is both $(rq)_*B$ and r_*q_*B. One should check that $\varepsilon \circ a$ coincides with the counit $(rq)^*(rq)_*B \to B$.

Now one way of rewriting is

$$r_* \circ q_* \circ b_! \simeq w_! \circ (ms)_* \circ (\varepsilon a)^*$$

with just one application of distributivity.

Another way of rewriting is

$$r_* \circ q_* \circ b_! \simeq r_* \circ c_! \circ g_* \circ \varepsilon^* \simeq w_! \circ m_* \circ e^* \circ g_* \circ \varepsilon^* \simeq w_! \circ m_* \circ s_* \circ a^* \circ \varepsilon^*$$

The two end results are clearly isomorphic by two applications of the compatibility-with-composition rule. But how do we know that these two way of arriving here coincide?

The answer is to take all the involved 2-cells and break them up according to their construction. We know that all the 2-cells we use are built up from units and counits (and compatibility-with-composition triangles)—and their inverse when they exist.

When we tackle this problem, we don't really care whether the distributivity 2-cell is invertible. In fact the proof goes through for any diagram of the shape, if just we are careful about the direction of the transformations.

We can deduce the equality of the two composite 2-cells by first studying the similar question for the 2-cell of Lemma 8.3.2.

8.5 Composition

Assuming we have an intrinsic characterisation of polynomial functors then it should follow readily from stability properties of the conditions in the characterisation that the composite of two polynomial functors is again polynomial. Such a proof would not support our viewpoint that all operations on polynomial functors should take place on the representing sets and maps.

thm:composite

8.5.1 Lemma. *The composite of two polynomial functors is again polynomial.*

Proof. Consider the beautiful diagram

$$\begin{tikzcd}
&&&&&& \cdot \arrow[rrr, "\Pi"] \arrow[dll, "\Delta"'] &&& \cdot \arrow[rr, "\Pi"] \arrow[dll, "\Delta"] && \cdot \arrow[dd, "\Sigma"] \\
&&&& \cdot \arrow[rrr, "\Pi"'] \arrow[dll, "\Delta"'] &&& \cdot \arrow[dll, "\Delta"] \arrow[drr, "\Sigma"] \\
&& \cdot \arrow[rrr, "\Pi"'] \arrow[dll, "\Delta"'] &&& \cdot \arrow[drr, "\Sigma"'] &&&& \cdot \arrow[rr, "\Pi"'] \arrow[dll, "\Delta"] && \cdot \arrow[drr, "\Sigma"] \\
\cdot &&&&&&& \cdot &&&&&& \cdot
\end{tikzcd}$$

In this diagram the labels just indicate what sort of operation we are performing along the given map, Δ indicating pullback, Σ lowershriek, and Π lowerstar. Given a set X over the lower left-hand set, dragging X through the lower six maps is to apply two polynomial functors to it. According to the Beck-Chevalley conditions and distributivity, dragging X through the top sequence of arrows gives the same result, and doing this is clearly a polynomial functor. Indeed the three squares are defined by pullback, and the pentagon is constructed as in the distributivity result.

Alternatively, you can give each map a specific name and start rewriting using Beck-Chevalley conditions and distributivity. E.g. suppose the maps are called

$$\begin{tikzcd}
 & \cdot \arrow[dl, "a"'] \arrow[r, "b"] & \cdot \arrow[dr, "c"] & & \cdot \arrow[dl, "z"'] \arrow[r, "q"'] & \cdot \arrow[dr, "m"] & \\
\cdot & & & \cdot & & & \cdot
\end{tikzcd}$$

then the proof starts like this: form the pullback

$$\begin{tikzcd}
\cdot \arrow[r, "x"] \arrow[d, "k"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "z"] \\
\cdot \arrow[r, "c"'] & \cdot
\end{tikzcd}$$

and use Beck-Chevalley:

$$m_! \circ q_* \circ z^* \circ c_! \circ b_* \circ a^* \simeq m_! \circ q_* \circ x_! \circ k^* \circ b_* \circ a^*$$

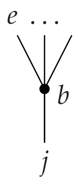
exchanging the middle lowershriek and upperstar. And so on, inventing names for the maps in the big diagram. (You do see it all in the diagram!) □

graphical

8.5.2 Graphical interpretation. The following graphical interpretation is very useful. Referring to a polynomial functor

$$\begin{tikzcd}
 & E \arrow[dl, "s"'] \arrow[r, "p"] & B \arrow[dr, "t"] & \\
I & & & J
\end{tikzcd} \tag{8.6}$$

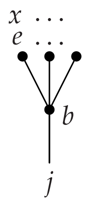
the important aspects of an element $b \in B$ are: the fibre $E_b = p^{-1}(b)$ and the element $j := t(b) \in J$. We capture these data by picturing b as a (non-planar) bouquet (also called a corolla)



Hence each leaf is labelled by an element $e \in E_b$, and each element of E_b occurs exactly once. In virtue of the map $s : E \to I$, each leaf $e \in E_b$ acquires furthermore an implicit decoration by an element in I, namely $s(e)$.

An element in E can be pictured as a bouquet of the same type, but with one of the leaves marked (this mark chooses the element $e \in E_b$, so this description is merely an expression of the natural identification $E = \coprod_{b \in B} E_b$). Then the map $p : E \to B$ consists in forgetting this mark, and s returns the I-decoration of the marked leaf.

Consider now a set over I, say $f : X \to I$. Then the elements of $P(X)$ are bouquets as above, but where each leaf is furthermore decorated by elements in X in a compatible way:



The compatibility condition for the decorations is that leaf e may have decoration x only if $f(x) = s(e)$. The set of such X-decorated bouquets is naturally a set over J via t (return the label of the root edge). More formally, $P(X)$ is the set over B (and hence over J via t) whose fibre over $b \in B$ is the set of commutative triangles

$$\begin{tikzcd}
X \arrow[dr, "f"'] & & E_b \arrow[ll] \arrow[dl, "s"] \\
& I. &
\end{tikzcd}$$

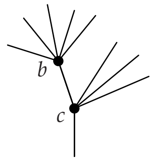
Let us work through the diagram and interpret all the sets in graphical terms. Suppose the polynomial functors are given by $I \leftarrow \overline{B} \to B \to J$ and

$J \leftarrow \overline{C} \rightarrow C \rightarrow K$

$$\begin{tikzcd}
& & & \overline{W} \arrow[rr] \arrow[dl] & & W \arrow[rr] \arrow[dl] & & q_*V = Q(B) \arrow[dd] & \\
& & \overline{V} \arrow[rr] \arrow[dl] & & V \arrow[dl] \arrow[dr] & & \text{distr.} & & \\
& \overline{B} \arrow[rr, "p"'] \arrow[dl] & & B \arrow[dr] & & \overline{C} \arrow[rr, "q"'] \arrow[dl] & & C \arrow[dr] & \\
I & & P & & J & & Q & & K
\end{tikzcd} \tag{8.7}$$

big-comp-diagram

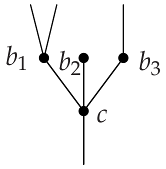
According to the general description, the set V is just $B \times_J \overline{C}$. This is the set of pairs of bouquets, one B-bouquet b and one C-bouquet f with a marked leaf, such that the decoration of the marked leaf of f is the same as the root decoration of b. In other words, the elements are two-node trees, the lower node being c and the upper node b, and all edges decorated in J:



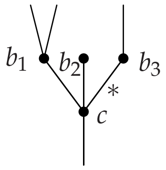
Now push this set forth along q: the set q_*V is described as follows, cf. the general description of pushing forth: its elements are B-bouquets, where furthermore each leaf is decorated by an element in T. Indeed, by the general description, q_*V is a set over C whose fibre over $c \in C$ is the set of commutative triangles

$$\begin{tikzcd}
B \arrow[dr] & & \overline{C}_c \arrow[ll, "\varphi"'] \arrow[dl] \\
& J. &
\end{tikzcd}$$

I.e. to each leaf of c we have to choose a B-bouquet with the output type equal to the input type at the leaf. In other words, we just have the configurations of all two-level trees, with the bottom level being a C-bouquet, and the top levels being B-bouquets.

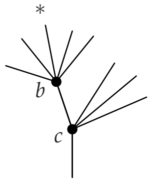


The set $W := q^*q_*V$ is described in a similar manner: they are just two-level trees with one of the inner edges marked. Indeed, formally its elements (in the fibre over $c \in C$) are pairs (f, φ) where $f \in \overline{C}_c$ and φ is a diagram like above, so f just selects an inner edge in the previous figure:

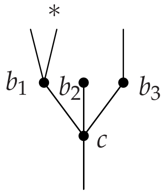


The map from here to V simply forgets about the upper nodes not corresponding to the marked inner edge. I.e. prune all non-marked edges. Formally, it is the evaluation map, sending a pair (f, φ) to the pair $(\varphi(f), f)$, i.e. we no longer have to select a B-bouquet for every leaf, but only for the one singled out by f.

Finally we have the pulled back versions of these sets. We denote by $\overline{V}$ the pullback of V along p. Clearly this is just the set of two-node trees (lower node in C and upper in B) with a marked leaf of the upper node.



Finally $\overline{W} := p^*W$ is the same configurations as in $W = q^*q_*V$, but with one leaf marked of the node lying over the marked inner edge. Note that with this leaf marked, the mark on the inner edge becomes superfluous since it is determined uniquely as the inner edge leading to the marked leaf. So in conclusion, $\overline{W}$ is the set of two-level trees with one marked leaf.



Altogether we just recover the description we arrived at by heuristic arguments, namely that the composite is defined in terms of two-level trees.

In particular, the base set for the composite is seen to be

$$q_*(B \times_J C)$$

which is also what we knew from some other computation we did earlier.

This is not strictly associative, because the involved operations, pullback, pushforth, and so on are only well-defined up to unique isomorphism. If we chose them as algebraic operations from the beginning then we get a well-defined composition law, but it will only be associative up to isomorphism. It is clear that the pentagon equation is satisfied because all the weakness comes from universal properties.

Another proof of coherence: given a triple composition, the associator will be a natural transformation between two slightly different polynomial functors. However, each of these polynomial functors has a natural isomorphism to the triple composition of functors, which is strictly associative, and in fact the associator is defined through these, and since everything is thus anchored in the strict world of functors, coherence is automatic.

Rewrite systems and coherence

The following is an easy consequence of the previous subsection:

8.5.3 Corollary. *The class of polynomial functors is the smallest class of functors between slices of* ***Set*** *containing the pullback functors and their adjoints, and closed under composition and natural isomorphism.* □

We often use a Beck-Chevalley isomorphism as if it were the identity. And we treat distributivity in the same careless way. But we ought to worry about coherence. . .

Let's define more generally a polynomial functor to be any (finite) composite of upperstar, lowerstar, and lowershriek functors. Each of these classes of functors are in particular polynomial, and since we already showed that composites of polynomial functors are again polynomial, it is clear that this new definition is not more general than the original: We will see in a moment that every such composite can be brought on 'normal form' as the usual three-step polynomial functor we use all the time.

However, it is interesting to analyse these rewriting processes more carefully.

8.5.4 Proposition. *Every composite of upperstar, lowerstar, and lowershriek functors can be brought on normal form:*

$$\begin{tikzcd}
& E \arrow[r, "p"] \arrow[dl, "s"'] & B \arrow[dr, "t"] & \\
I & & & J
\end{tikzcd} \tag{8.8}$$

$P : \mathbf{Set}/I \to \mathbf{Set}/J$ defined by

$$\mathbf{Set}/I \xrightarrow{s^*} \mathbf{Set}/E \xrightarrow{p_*} \mathbf{Set}/B \xrightarrow{t_!} \mathbf{Set}/J.$$

Proof. First we move all the lowershriek functors to the end: given some occurrence

$$g^* \circ f_!$$

as in the diagram

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

complete to a pullback diagram

$$\begin{tikzcd}
\cdot \arrow[r, "k"] \arrow[d, "h"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "g"] \\
\cdot \arrow[r, "f"'] & \cdot
\end{tikzcd}$$

Then we have $g^* \circ f_! = k_! \circ h^*$, by Beck-Chevalley.

In the situation where we have $p_* \circ a_!$ as in the diagram

$$\begin{tikzcd}
X \arrow[d, "a"'] & \\
\cdot \arrow[r, "p"'] & \cdot
\end{tikzcd}$$

complete to a pullback diagram diagram

$$\begin{tikzcd}
p^* p_* X \arrow[r, "q"] \arrow[d, "\varepsilon"'] \arrow[rdd, phantom, "\lrcorner", very near start] & p_* X \arrow[dd, "g"] \\
X \arrow[d, "a"'] & \\
\cdot \arrow[r, "p"'] & \cdot
\end{tikzcd}$$

and invoke distributivity:

$$p_* \circ a_! = g_! \circ q_* \circ \varepsilon^*$$

In conclusion we can always move all occurrences of lowershriek to the end.

Once we have done that, we can move all the occurrences of upperstar to the beginning. We just ave to move them past all the occurrences of lowerstar, and we can do that by Beck-Chevalley. □

We may want to formulate this as a double factorisation system. Assuming we have an abstract characterisation of polynomial functors, say first that every polynomial functor factorises uniquely as a right adjoint (the upperstar-lowerstar part) followed by a ???-part (characterising the lowershriek part). Next, show that every right adjoint polynomial functor factorises uniquely as an upperstar followed by a lowerstar. (There is no factorisation system in which the left-hand class is the class of upperstars, because the class of lowershriek-and-lowerstar functors is not closed under composition, cf. distributivity.)

Here is the issue: we have certain rewriting rules that allow us to start from any composite of upperstars, lowershrieks and lowerstars, and rewrite to arrive at the normal form. The rules are: move the lowershrieks towards the right, swapping with upperstars using Beck-Chevalley, and swapping with lowerstars using distributivity (this step produces a new upperstar). We can do this with all the lowershrieks until they are all concentrated in the end. Then we can interchange upperstars and lowerstars using Beck-Chevalley until finally we have all the upperstars in the beginning. Then we can compose all the upperstars, compose all the lowerstars, and compose all the lowershrieks.

But we could also compose two adjacent terms of the same type whenever we can. How do we know that the various isomorphisms such obtained from the initial expression and to the normal form are the same? Can we even guarantee the normal forms to be the same independently of how we rewrite? Yes, this is OK, because all steps are isomorphisms, so the resulting normal forms are always isomorphic. But could it be that there is more than one isomorphism?

THE RESULTS IN SECTION 8.4 SHOW THAT EVERYTHING IS OK. . .

What are the gymnastics results for distributivity?

Alternatively, perhaps we can *deduce* the coherence result from the biequivalence with the 2-category of polynomial functors, established in the coming chapters. . .

8.6 Basic properties

Sec:many:basic

8.6.1 Lemma. *Polynomial functors (over an arbitrary lccc) preserve connected limits.*

Proof. Indeed, p_* and s^* are right adjoints so they preserve all limits, and $t_!$ preserves connected limits by Lemma 8.2.5.

□

In particular polynomial functors are cartesian, and since monos can be characterised in terms of pullbacks, it follows that polynomial functors preserve monos.

8.6.2 Intrinsic characterisations of polynomial functors. Here we give some intrinsic characterisations of polynomial functors over ***Set***. This case is somewhat special due to the equivalence $\boldsymbol{Set}/I \simeq \boldsymbol{Set}^I$.

thm:six **8.6.3 Theorem.** *For a functor $P : \textbf{Set}/I \to \textbf{Set}/J$, the following conditions are equivalent.*

item:poly *(i) P is polynomial.*

item:connected *(ii) P preserves connected limits (or, equivalently, pullbacks and cofiltered limits, or equivalently, wide pullbacks).*

item:famrepr *(iii) P is familially representable (i.e. a sum of representables).*

item:topos *(iv) The comma category $(\textbf{Set}/J)\downarrow P$ is a presheaf topos.*

item:lra *(v) P is a local right adjoint (i.e. the slices of P are right adjoints).*

item:generic *(vi) P admits strict generic factorisations [103].*

tem:normalform *(vii) Every slice of $\mathrm{el}(P)$ has an initial object (Girard's normal-form property).*

The equivalences (ii) ⇔ (v) ⇔ (vi) go back to Lamarche [67] and Taylor [102], who were motivated by the work of Girard [43], cf. below. They arrived at condition (vi) as the proper generalisation of (vii), itself a categorical reformulation of Girard's normal-form condition [43]. Below we give a direct proof of (i) ⇔ (vii), to illuminate the relation with Girard's normal functors. The equivalence (ii) ⇔ (iii) is due to Diers [32], and was clarified further by Carboni and Johnstone [25], who established in particular the equivalence (ii) ⇔ (iv) as part of their treatment of Artin gluing. The equivalence (i) ⇔ (iii) is also implicit in their work, the one-variable case explicit. The equivalence (i) ⇔ (v) was observed by Weber [104], who also notes that on general presheaf toposes, local right adjoints need not be polynomial: for example the free-category monad on the category of directed graphs is a local right adjoint but not a polynomial functor.

8.6.4 Finitary functors. A polynomial functor $P : \textbf{Set}/I \to \textbf{Set}/J$ is *finitary* if it preserves filtered colimits. If P is represented by $I \leftarrow B \to A \to J$, this condition is equivalent to the map $B \to A$ having finite fibres.

8.6.5 Girard's normal functors. Girard [43], aiming at constructing models for lambda calculus, introduced the notion of *normal functor*: it is a functor $\textbf{Set}^I \to \textbf{Set}^J$ which preserves pullbacks, cofiltered limits and filtered colimits, i.e. a finitary polynomial functor. Girard's interest was a certain normal-form property (reminiscent of Cantor's normal form for ordinals), which in modern language is (vii) above: the normal forms of the functor are the initial objects of the slices of its category of elements. Girard, independently of [55], also proved that these functors admit a power series expansion, which is just the associated (flat) analytic functor. From Girard's proof we can extract in fact a direct equivalence between (i) and (vii) (independent of the finiteness condition). The proof shows that, in a sense, the polynomial representation *is* the normal form. For simplicity we treat only the one-variable case.

thm:normalform **8.6.6 Proposition.** *A functor P : **Set** $\to$ **Set** is polynomial if and only if every slice of* $\mathrm{el}(P)$ *has an initial object.*

Proof. Suppose P is polynomial, represented by $B \to A$. An element of P is a triple (X, a, s), where X is a set, $a \in A$, and $s : B_a \to X$. The set of connected components of $\mathrm{el}(P)$ is in bijection with the set $P(1) = A$. For each element $a \in A = P(1)$, it is clear that the triple $(B_a, a, \mathrm{id}\, B_a)$ is an initial object of the slice $\mathrm{el}(P)/(1, a, u)$, where u is the map to the terminal object. These initial objects induce initial objects in all the slices, since every element (X, a, s) has a unique map to $(1, a, u)$.

Conversely, suppose every slice of $\mathrm{el}(P)$ has an initial object; again we only need the initial objects of the special slices $\mathrm{el}(P)/(1, a, u)$, for $a \in P(1)$. Put $A = P(1)$. It remains to construct B over A and show that the resulting polynomial functor is isomorphic to P. Denote by (B_a, b) the initial object of $\mathrm{el}(P)/(1, a, u)$. Let now X be any set. The unique map $X \to 1$ induces $P(X) \to P(1) = A$, and we denote by $P(X)_a$ the preimage of a. For each element $x \in P(X)_a$, the pair (X, x) is therefore an object of the slice $\mathrm{el}(P)/(1, a, u)$, so by initiality we get a map $B_a \to X$. Conversely, given any map $\alpha : B_a \to X$, define x to be the image under $P(\alpha)$ of the element b; clearly $x \in P(X)_a$. These two constructions are easily checked to be inverse to each other, establishing a bijection $P(X)_a \cong X^{B_a}$. These bijections are clearly natural in X, and since $P(X) = \sum_{a \in A} P(X)_a$ we conclude that P is isomorphic to the polynomial functor represented by the projection map $\sum_{a \in A} B^a \to A$. □

8.7 Examples

Special classes of polynomial functors:

8.7.1 Constant polynomial functors. The constant polynomial functors $\textbf{\textit{Set}}/I \to \textbf{\textit{Set}}$ with value Λ are represented by the diagrams $I \leftarrow \emptyset \to \Lambda$. More generally the constant functors $\textbf{\textit{Set}}/I \to \textbf{\textit{Set}}/J$ with value Λ/J are represented by $I \leftarrow \emptyset \to \Lambda \to J$. If we furthermore want it to be equal in each output component, we should take a fixed set Λ and use $\Lambda \times J$.

This defines a functor $\textbf{\textit{Set}}/J \longrightarrow \textbf{\textit{PolyFun}}(I,J)$ sending Λ/J to the constant functor. Check that this one has a right adjoint which is evaluation at $\emptyset$. We can compose this adjunction with the adjunction $\textbf{\textit{Set}} \to \textbf{\textit{Set}}/J$ which sends Λ to $\Lambda \times J$. (The right adjoint is the forgetful functor.) This gives a functor $\textbf{\textit{Set}} \to \textbf{\textit{PolyFun}}(I,J)$ which has as right adjoint evaluation at the empty set.

multXi **8.7.2 Partial identity functors (one for each type).** WAIT UNTIL WE HAVE INTRODUCED MULTIPLICATION OF POLYNOMIA FUNCTORS. For each $i \in I$ we have the identity functor $X/I \mapsto X_i$. It is represented by the diagram $I \xleftarrow{i} 1 \to 1$. The effect of multiplying with this functor which de denote by X_i is:

$$X_i \cdot (I \leftarrow E \to B) = I \leftarrow E + B \to B$$

where the new B-summand in the top space maps to i. In other words, to every fibre there is added an extra element (of type i).

8.7.3 Linear functors. This means that the middle map (the multiplication part) is a bijection—we can assume it is the identity map. Then we are talking about diagrams like

$$\begin{tikzcd} & A \arrow[dl, "s"'] \arrow[dr, "t"] & \\ I & & J \end{tikzcd}$$

The set A is indexed in simultaneously by the two sets I and J, so it is really a matrix of sets! Now many concepts from linear algebra carry over, but of course it is more precisely linear algebra over $\mathbb{N}$ we are talking about! Since this is cool anyway, the whole next subsection is dedicated to this...

affine-many **8.7.4 Affine functors.**

8.7.5 In case of map over I. Suppose we are given a commutative diagram

$$\begin{tikzcd}
E \arrow[rr, "p"] \arrow[dr, "s"'] & & B \arrow[dl, "t"] \\
& I &
\end{tikzcd}$$

This is a special case of a polynomial functor. In this case, the formula

$$\sum_{i \in I} \sum_{b \in B_i} \prod_{e \in E_b} X_{s(e)}$$

simplifies a little bit: since the triangle commutes we have $s(e) = i$, so the expression simplifies to

$$\sum_{i \in I} \sum_{b \in B_i} X_i^{E_b}.$$

Taking for granted the statement that polynomial functors make sense in any topos, then the functors of this type are those that are one-variable polynomial functors in the topos ***Set***$/I$.

The free-category functor on graphs with fixed object set

Sub:free-cat

Consider the set of linear functors with endpoints S. This is the same thing as a (non-reflexive) directed graph with vertex set S. We now construct the free category on this graph. This will be the left adjoint to the forgetful functor from categories with object set S to directed graphs with vertex set S. Put differently it is the adjunction between linear polynomial monads with endpoint set S, and linear polynomial functors with endpoint set S. We are talking about endofunctors on ***Set***$/(S \times S)$.

Given a graphs with vertex set S there is a free category on that graph, and of course it has object set S again. The corresponding monad on S-graphs is polynomial. A graph with vertex set S is a diagram $E \rightrightarrows S$, so the category of graphs with vertex set S is the slice category ***Set***$/(S \times S)$.

The terminal graph with vertex set S is the chaotic graph on S (i.e. there is precisely one edge between any two vertices). That's just $S \times S \rightrightarrows S$. The free category on that graph is this: it has the set S as object set, and it has the set of paths in the chaotic graph on S as arrow set. We do not allow the empty path. A path is therefore given by specifying a non-empty chain of elements of S. The shortest paths are those with only one element of S (then the corresponding arrow will be the identity arrow we introduce for that element). So $T1 = S^{+}$. This set has two projections to S, namely returning the first and the last element of the chain.

For a general graph with vertex set S, that's $X \to S \times S$, we need rather to specify a sequence

```
      x_1     x_2     x_3     ...     x_n
  s_0     s_1     s_2                     s_n
```

where x_i is an edge from s_{i-1} to s_i.

This assignment is polynomial: the E-set is the set of non-empty chains of elements in S with two consecutive elements marked. In the end the polynomial functor is given by

$$\begin{tikzcd}
& S^{+\prime\prime} \arrow[r] \arrow[dl] & S^{+} \arrow[dr] & \\
S \times S & & & S \times S
\end{tikzcd}$$

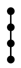
Indeed, given a graph with vertex set S and edge set X, that is, $X \to s \times S$, then by pulling back to $S^{+\prime\prime}$ and pushing forth to S^{+} we get the set of alternating incidence sequences as required.

So the free category on $X \to S \times S$ is the category with arrow set

$$\sum_{n \in \mathbb{N}} X \times_S \cdots \times_S X$$

and with the two outer projections.

You can also say that it is the set of



with dots decorated in S and edges decorated in X, subject to the obvious compatibility requirement.

Now the set S^{+} is already the set of such linear graphs with nodes decorated by elements in S. So all the polynomial functor does is to further decorate the linear graph with X at the nodes, in a compatible way.

We shall later come back to the free-category monad without fixing the vertex set. This monad is *not* polynomial in the sense of this chapter. It is polynomial in a more general sense, where we use presheaf categories instead of slice categories.

Then instead of just having the S-decorations fixed in advance, and let the polynomial functor provide the X-decorations, we will have abstract linear graphs and let the polynomial functor (in the generalised sense) decorate everything.

Chapter 9

Examples

9.1 Linear functors (matrices)

Sec:linear

9.1.1 Notation for linear algebra. We are going to develop some rudimentary linear algebra, and let us fix some notation. We want the following: composition is denoted left-to-right, so that the composite of linear maps

$$\begin{tikzcd} U \arrow[r, "A"] & V \arrow[r, "B"] & W \end{tikzcd}$$

will be the matrix product AB. Hence in particular, a linear maps does $\mathbf{x} \mapsto \mathbf{x} \cdot A$. This means that vectors are row vectors.

We want to distinguish upper and lower indices like in physics. Hence the above products should read

$$\sum_j A^i_j \cdot B^j_k = C^i_k$$

and $\sum_i x_i A^i_j$

If you think this looks strange, try to work out a few matrix multiplications, just to get used to it before starting on the linear functors...

An alternative notation would be to move the upper indices down on the left side. Then matrix multiplication looks like this

$${}_iA_j \cdot {}_jB_k = {}_iC_k$$

9.1.2 Linear functors. This means that the middle map (the multiplication part) is a bijection. For simplicity we assume it is the identity map, so we are talking about diagrams

$$\begin{tikzcd} & A \arrow[dl, "s"'] \arrow[dr, "t"] & \\ I & & J \end{tikzcd}$$

The corresponding polynomial functor F is

$$\begin{array}{rcl} \boldsymbol{Set}/I & \longrightarrow & \boldsymbol{Set}/J \\ (X_i \mid i \in I) & \longmapsto & (\sum_{a \in A_j} X_{s(a)} \mid j \in J) \end{array}$$

which of course we call a *linear functor*. As usual, let A_j denote the subset $t^{-1}(j) \subset A$. Since there are no exponents involved in linear functor, we will allow ourselves to use upper indices now: let A^i denote the subset $s^{-1}(i) \subset A$. Let A^i_j denote the intersection $A^i \cap A_j$. Hence

$$A = \sum_{i,j} A^i_j.$$

Now, this is just a matrix of sets—the rows of the matrix are indexed by I, and the columns are indexed by J. Many notions like rows and columns usually assume that I and J are finite ordered sets, but in fact many concepts don't require this assumption.

Now the linear functor is

$$\begin{array}{rcl} F : \boldsymbol{Set}/I & \longrightarrow & \boldsymbol{Set}/J \\ X & \longmapsto & A \times_I X \\ (X_i \mid i \in I) & \longmapsto & (Y_j \mid j \in J) \end{array}$$

with

$$Y_j = A^i_j X_i$$

WITH THE ABOVE CONVENTION, I THINK THIS SOULD BE $X \times_I A$ and $X_i \times A^i_j$

So the linear functors $\boldsymbol{Set}/I \to \boldsymbol{Set}/J$ are given by the set $\boldsymbol{Set}/(I \times J)$.

9.1.3 Composition of linear functors—matrix multiplication. Given linear functors (and we have to find a good convention to make the order look right in all cases. . .)

$$I \leftarrow A \rightarrow J \leftarrow B \rightarrow K$$

clearly the composite linear functor is given by the fibre product $A \times_J B$. Let us look at the ${}^i_k$ fibre $(A \times_J B)^i_k$. It is the sum

$$\sum_{j \in J} A^i_j \times B^j_k$$

which says that $A \times_J B$ is precisely the matrix product of A and B. Well, what else would you expect?

In conclusion, the formula for multiplying two matrices is just a pullback formula!

The linear functors $I \leftarrow B \rightarrow J$ sit inside $\textbf{\textit{PolyFun}}(I,J)$ as a full subcategory.

UPS: WE HAVEN'T YET INTRODUCED THOSE CATS

Note that for between linear functors there is no difference between general natural transformation and cartesian ones, so $\textbf{\textit{Lin}}(I,J)$ is also a full subcategory of $\textbf{\textit{PolyFun}}(I,J)$. However, the following result only applies to $\textbf{\textit{Poly}}^{\mathrm{c}}(I,J)$:

9.1.4 Lemma. *The inclusion functor* $\textbf{Lin}(I,J) \rightarrow \textbf{Poly}^{c}(I,J)$ *has a right adjoint.*

It is the functor

$$\begin{array}{rcl} \textbf{\textit{Poly}}^{\mathrm{c}}(I,J) & \longrightarrow & \textbf{\textit{Lin}}(I,J) \\ E/B & \longmapsto & E_1/B_1 \end{array}$$

which only retains the linear part. I.e. we take the subset $B_1 \subset B$ consisting of elements whose fibre is singleton and then $E_1 \simeq B_1$.

Indeed if $I \leftarrow M \rightarrow J$ is a linear functor, and $I \leftarrow E \rightarrow B \rightarrow J$ is a general polynomial functor, then to give a cartesian map from M/M into E/B is the same as giving one into the linear part, because all elements in M are of arity 1, so have to map to the arity-1 operations in B.

Note that in general there is no cartesian map from E/B to M/M, so there is certainly not a left adjoint to the inclusion.

Note that the linear-part functor is neither full not faithful: any two cartesian maps between polynomial functors without linear part are mapped to the identity map of $\emptyset$, so linear-part functor is not faithful. It is not full either, because there are no cartesian maps between the polynomial functors $3/1$ and $2/1$, yet there is the identity map on the linear part.

THE FOLLOWING DISCUSSION SHOULD BELONG TO THE VARIABLE-TYPE CASE: HOWEVER: if we restrict to the category of trees (the big one with three sorts of maps), then we know that a map between trees is completely determined by its value on edges (these are unary operations), so this shows in particular that the linear-part functor is faithful when restricted to the subcategory of trees. It is still not full: for the single-dot tree of arity 2, then the linear part is the functor $3 \leftarrow 3 \rightarrow 3$. This one has 3! automorphisms whereas the original tree only has 2.

More generally, define $\boldsymbol{Poly}^{\mathrm{c}}_n$ to be the sub-2-category of $\boldsymbol{Poly}^{\mathrm{c}}$ consisting of the polynomial functors of degree at most n. This means that the cardinality of each fibre of $E \rightarrow B$ is at most n. (We could define $\boldsymbol{Aff} := \boldsymbol{Poly}^{\mathrm{c}}_1$, the affine functors mentioned in (1.2.4 and) 8.7.4.) Again the inclusion functor $\boldsymbol{Poly}^{\mathrm{c}}_n \hookrightarrow \boldsymbol{Poly}^{\mathrm{c}}$ has a right adjoint which only takes the subset of B for which the fibre is at most n.

An important case of this is the $n = 0$: the category $\boldsymbol{Poly}^{\mathrm{c}}_0(1,1)$ is naturally identified with $\boldsymbol{Set}$, and more generally, $\boldsymbol{Poly}^{\mathrm{c}}_0(I,J)$ is identified with $\boldsymbol{Set}/J$. The right adjoint is evaluation at $\emptyset$.

And similarly, for any $m \leq n$, the inclusion $\boldsymbol{Poly}^{\mathrm{c}}_m \hookrightarrow \boldsymbol{Poly}^{\mathrm{c}}_n$ has a right adjoint.

Similarly, we could consider polynomial functors of precisely degree n, or perhaps even go for those of degree F for a fixed set F.

9.1.5 Square matrices. Square matrices are linear endofunctors $\boldsymbol{Set}/I \rightarrow \boldsymbol{Set}/I$. They are given by a pair of maps

$$I \leftarrow A \rightarrow I$$

which is to say they are graphs! (I.e. nonreflexive directed graphs.)

lin-monad **9.1.6 Linear monads = small categories.** In particular we have linear monads. These are precisely small categories!

An important class of linear monads are given by products: given a map $\alpha : I \to J$, then we can form the product:

$$\begin{tikzcd}
 & I \times_J I \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & \\
I \arrow[dr, "\alpha"'] & & I \arrow[dl, "\alpha"] \\
 & J &
\end{tikzcd}$$

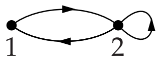
The linear endofunctor defined by this span is

$$\pi_{2*} \circ \pi_1^* \;\; = \;\; \alpha^* \circ \alpha_!$$

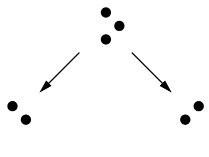
(via Beck-Chevalley). Since this endofunctor is induced by an adjoint pair, it is a monad! (cartesian as all linear monads). As a monad it is given by $X/I \longmapsto X \times_J I$. The monad structure is $X \times_J I \times_J I \to X \times_J I$ remove the middle factor. The unit is given by $X \mapsto X \times_J I$ (remembering that X was an object over I).

Since it is a linear monad, it is a category! It has object set I. The set of arrows is $I \times_J I$. There is an arrow from i_0 to i_1 if and only if they have the same image in J. Hence the category is a groupoid, and even a codiscrete groupoid: it is the equivalence relation generated by the map α.

9.1.7 Fibonacci sets. Consider the directed graph $E \rightrightarrows V$ with two vertices and three edges: one in each direction between the vertices, and one loop at one of the vertices:



This is a span



This is the polynomial functor sending

$$(X_1, X_2) \longmapsto (X_2, X_1 + X_2)$$

If we give it as input the sets $(0,1)$, and then iterate, and then pick out the first coordinate, we get the sequence of sets

$$0,1,1,2,3,5,8$$

the Fibonacci sets!

We can also look just at the sequence E^n, where we interpret the set of edges as a matrix. We get

$$E^0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E^1 = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad E^2 = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}, \quad E^3 = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix},$$

$$E^4 = \begin{pmatrix} 2 & 3 \\ 3 & 5 \end{pmatrix}, \quad E^5 = \begin{pmatrix} 3 & 5 \\ 5 & 8 \end{pmatrix}, \quad E^6 = \begin{pmatrix} 5 & 8 \\ 8 & 13 \end{pmatrix}, \quad E^7 = \begin{pmatrix} 8 & 13 \\ 13 & 21 \end{pmatrix},$$

9.1.8 Traces. The *trace* of a graph is by definition the subgraph with the same vertices but only the loops as edges. The reason for the terminology is of course that if we consider the graph as a matrix, then the subgraph of its loops are the diagonal entries of the matrix. Maybe we should rather say that the trace of a graph is the set of loops. That's the sum of all the diagonal sets.

The *Lucas sets* are the sets $(\mathrm{Tr}(M^n) \mid n \in \mathbb{N})$, where M is the Fibonacci graph.

9.1.9 Graph theory in terms of linear algebra. There are many notions from graph theory that allow an elegant formulation in terms of linear algebra. Interpreting a graph as a matrix is nothing but the familiar adjacency matrix from graph theory, just with sets instead of numbers. . .

9.1.10 The free category on a directed graph. Consider the graph given by the matrix G^n, where G is a graph, and $n \in \mathbb{N}$. This graph is the graph of paths in G of length n. In other words,

$$\sum_{n \in \mathbb{N}} G^n$$

is the free category on the graph G. Note that the sum is sum of matrices, not a sum of graphs. This means that the sum, if interpreted as a graph, still has the same set of vertices. The sum of matrices is of course just the entry-wise sum. This is also the sum of linear functors, which in turn is induced from the sum in the category $\boldsymbol{Set}/V$.

Note the similarity with the free-monoid monad on $\boldsymbol{Set}$, the familiar $X \mapsto \sum_{n\in\mathbb{N}} X^n$. This is indeed a special case, namely where we interpret the set X as the graph $X \rightrightarrows 1$ (i.e. X is the set of edges in a graph with only one vertex). Then matrix multiplication is just cartesian product of sets, so the two formulae agree.

There is a slogan here: *A set is a graph with only one vertex.*

Let G be the Fibonacci graph. It makes sense to call the free category on G the Fibonacci category – why not?

9.1.11 Some sort of Perron-Frobenius theorem? The classical Perron-Frobenius theorem states that a square matrix with non-negative entries (assumed to be an irreducible matrix, i.e. not similar to an upper-triangular block matrix) has a maximal real positive eigenvalue, and an associated eigenvector with nonnegative entries. In the categorified setting of linear functors, the theorem should be (at least the existence of an eigenset): given a graph G, there exists a set $X \to V$ (i.e. a family of sets indexed by the vertices and an eigenset Λ such that $X \cdot G = X \times \Lambda$. Now this is surely to hope too much, because in the setting of numbers, typically such an eigenvalue is not an integer. So we should modify the notion of eigenvector a little bit: an eigenvector is a vector of sets $X \to V$ such that X and $X \cdot G$ are proportional. This means that there are sets Λ_+ and Λ_- and a natural bijection

$$X \times \Lambda_+ \simeq X \times \Lambda_- \cdot G$$

But this is still not likely, because typically eigenvalues are irrational! Some problem to crack here!

But for some particular matrices, there should be some nice things to say. For example if the graph is d-regular, meaning that every vertex is of valence d, then d should be an eigenset, with eigenvector $(1, \ldots, 1)$. How to prove this?

9.2 Finite polynomials: the Lawvere theory of comm. semirings

9.2.1 Finite polynomials. Call a polynomial over $\boldsymbol{Set}$

$$I \leftarrow B \rightarrow A \rightarrow J \tag{9.1}$$

equ:polytamb

finite if the four involved sets are finite. Clearly the composite of two finite polynomials is again finite. For the purpose of this section, we identity two finite polynomials if they are isomorphic. Let $\mathbb{T}$ denote the skeleton of the category whose objects are finite sets and whose morphisms are finite polynomials (up to isomorphism). The main result of this section is the following.

9.2.2 Theorem. *The category $\mathbb{T}$ is the Lawvere theory for commutative semirings.*

para:tambara

9.2.3 Tambara's work. The category $\mathbb{T}$ was studied by Tambara [101], in fact in the more general context of finite G-sets, for G a finite group. His paper is probably the first to display and give emphasis to diagrams like (9.1). Tambara was motivated by representation theory and group cohomology, where the three operations Δ, Σ, Π are, respectively, 'restriction', 'trace' (additive transfer), and 'norm' (multiplicative transfer). Further study of *Tambara functors* was carried out by Brun [23]. We shall not go into the G-invariant achievements of [101], but wish to point out that the Theorem is implicit in Tambara's paper and should be attributed to him.

Lawvere theories

THIS SECTION NEEDS AN OVERHAUL: IT SHOULD PROBABLY BE MOVED TO AN INDEPENDENT SECTION AND PUT MORE EMPHASIS ON THE COMPARISON BETWEEN THEORIES, MONADS, AND OPERADS. AND STRESS THE DIFFERENCE BETWEEN FUNCTORIALITY W.R.T. BIJECTIONS (OPERADS) AND FUNCTORIALITY W.R.T. ALL MAPS (THEORIES). PERHAPS THIS SHOULD BE TREATED ALSO TOGETHER WITH SPECIES AND ANALYTICAL FUNCTORS.

Some good points to copy from [49].

Let us remind ourselves about Lawvere theories. Since this notion is very closely related to that of monad, we take the opportunity to explain that relationship.

9.2.4 Idea. An 'algebraic theory' in the naive sense, like the theory of groups, is something like this: a group is a set G with some operations and equations: the operations could be taken to be the composition law (that's a binary operation), the neutral element (that's a nullary operation, i.e. a constant) and inversion (that's an unary operation). And then there are the familiar equations that these operations must satisfy, e.g. associativity.

One observes that this definition makes sense not just in ***Set*** but in any category with finite products. Then the equations take form of commutative diagrams. However, it is possible to define what a group is in terms of just one binary operation, $(a,b) \mapsto ab^{-1}$, satisfying some complicated equation. Both of these are an 'equational theory' of groups. So what is the invariant content of this? Lawvere, in his thesis [71] from 1963, which is a milestone in the history of category theory, figured out the following: the theory is itself a category Θ with finite products, containing an object 1 which is the generic group (or whatever algebraic structure we are talking about): this means that every group in the naive sense arises uniquely as the image of 1 under a product-preserving functor $\Theta \to$ ***Set***. (The old-style theories are then interpreted as presentations of this category Θ, and more than one presentation may be possible.)

9.2.5 Lawvere theories. An *algebraic theory* in the sense of Lawvere, nowadays often called a *Lawvere theory*, is a category Θ with one object $\underline{n}$ for each natural number $n \in \mathbb{N}$, and having all finite products according to the rule

$$\underline{m} \times \underline{n} = \underline{m+n}$$

The morphisms are not explicitly referred to in the abstract definition, but of course the notion of product depends heavily on which morphisms exist in the category! In fact certain morphisms must exist, namely projections and diagonals. The various possibilities for introducing morphisms between the objects correspond to the algebraic structure encoded, be it groups, semirings, or whatever.

9.2.6 Models. An *model* for a Lawvere theory Θ, also called a Θ-*algebra*, is a product-preserving functor $\Theta \to$ ***Set***. A *homomorphism* of Θ-models is a natural transformation. (Naturality implies that in fact these homomorphisms are automatically compatible with products.)

[Historical remarks? relation with the older notion of clone (Hall).]

9.2.7 Hom sets of a theory. To describe the hom set $\Theta(\underline{m},\underline{n})$, observe that since $\underline{n}$ is the n-fold product of $\underline{1}$, we have

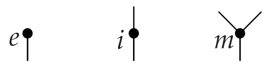
$$\Theta(\underline{m},\underline{n}) = \Theta(\underline{m},\prod_{i=1}^{n}\underline{1}) = \prod_{i=1}^{n}\Theta(\underline{m},\underline{1}),$$

so it is enough to describe the hom sets

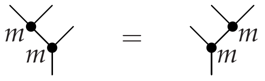
$$\Theta(\underline{m}, \underline{1})$$

which we refer to as the set of m-ary operations of the theory.

9.2.8 Example. In the theory of groups, in the usual presentation, we have the following generating operations: the composition law m (arity 2), the neutral element e (arity 0), and the inversion i (arity 1). But these are not all the operations of the theory: any combination of these three operations is again an operation (and some of these combinations are identified by the defining equations of the theory). For example we have a binary operation $(a, b) \mapsto ab^{-1}$ obtained by combining the composition law with inversion. The possible operations can be enumerated in terms of trees: picture the generating operations as bouquets according to their arity:



Then the set of all operations is the set of all planar trees made out of these three basic building blocks, modulo the equations. For example associativity says that



EXERCISE: write down all the possible operations of arity 4.

Given a theory in the sense of universal algebra, say the theory of groups, what is the corresponding Lawvere theory? For the sake of illustration, let us work with groups. The starting point is the diagram

$$\begin{tikzcd}
K \arrow[r] & \boldsymbol{Grp} \arrow[d, shift left=2, "U"] \arrow[d, phantom, "\dashv"] \\
\mathbb{F} \arrow[u] \arrow[r] & \boldsymbol{Set}. \arrow[u, shift left=2, "F"]
\end{tikzcd}$$

Here $F \dashv U$ is the free-forgetful adjunction, and $\mathbb{F}$ denotes a chosen skeleton of the category of finte sets and functions. To be explicit, choose symbols x_n for each $n \in \mathbb{N}$, and let the objects of $\mathbb{F}$ be $\mathbf{n} = \{x_1, \ldots, x_n\}$. The category K is the category of finitely generated free groups, or more explicitly, it is the full subcategory of ***Grp*** whose objects $F(\mathbf{n})$ are the free groups on the finite sets $\mathbf{n} = \{x_1, \ldots, x_n\}$.

The category K has finite sums since $F(\mathbf{m}) + F(\mathbf{n}) = F(\mathbf{m} + \mathbf{n})$ (and of course the inclusion $K \hookrightarrow$ ***Grp*** preserves sums). We now define the Lawvere theory to be $\Theta := K^{\mathrm{op}}$, and proceed to check that Θ-models are the same thing as groups in the naive sense.

Given a group G, we have the functor

$$\Theta = K^{\mathrm{op}} \longrightarrow \boldsymbol{Grp}^{\mathrm{op}} \xrightarrow{\mathrm{h}_G} \boldsymbol{Set}$$

where $\mathrm{h}_G = \boldsymbol{Grp}(-, G)$. This functor clearly preserves finite products, so it defines a Θ-model. To see that conversely every Θ-model defines a group is the interesting part.

The key point to note is that the object $\underline{1} = F(\mathbf{1})$ is a group object in Θ. To see this, let us show that it has the necessary operations and satisfies the required equations. Let us see how the operations $m : \underline{2} \to \underline{1}$ is defined. By construction of Θ we are talking about a map $F(\mathbf{1}) \to F(\mathbf{2})$, and by freeness, to give such a map is to give a single element in $F(\mathbf{2})$, the free group generated by x_1 and x_2. This element should be $x_1 x_2$, the product. We proceed similarly for each of the generating operation symbols in the theory: if the symbol is of arity k, we need a map $\underline{k} \to \underline{1}$ in Θ, that is, an element in $F(x_1, \ldots, x_k)$. Just take the value of the operation symbol on those k generators. Checking the axioms is similar. In conclusion, $\underline{1}$ is a group object. (More general conclusion: this argument works for any algebraic structure defined by finitary operation symbols and universally quantified equations.)

We can now easily show that each Θ-model defines a group, namely $G := A(\underline{1})$: since A preserves products, it sends group objects to group objects.

We have now argued that there is a one-to-one correspondence between groups and Θ-models. Of course a similar analysis is required to see that also group homomorphisms correspond to Θ-model homomorphisms, so that altogether we find an equivalence (in fact an isomorphism) of categories

$$\boldsymbol{Grp} \simeq \Theta\text{-}\boldsymbol{Mod}.$$

We should also note here that under this correspondence, the finitely generated free groups correspond exactly to the representable functors.

Here is another argument (HOW DOES IT FIT IN?): Every naive group defines a functor $\Theta \to \boldsymbol{Set}$, and this functor preserves finite products. Altogether, we find an embedding

$$\boldsymbol{Grp} \to \Theta\text{-}\boldsymbol{Mod} \subset [\Theta, \boldsymbol{Set}].$$

On the other hand, we saw that the Yoneda embedding $\Theta^{\mathrm{op}} \to [\Theta, \boldsymbol{Set}]$ factors through ***Grp***, and that in fact the essential image is the full subcategory finitely generated free groups. . .

The crucial point in this description was the notion of free group. Suppose now we are given a Lawvere theory Θ, and we don't yet know what it is a theory for. How to perform the above arguments? Well, we need first of all to establish a notion of free Θ-model. There is a forgetful functor $\Theta\text{-}\boldsymbol{Mod} \to \boldsymbol{Set}$ given by evaluation at $\underline{1}$. We need to show first that this functor always has a left adjoint. In fact:

9.2.9 Proposition. *For any Lawvere theory Θ, the category of models Θ-**Mod** is monadic over **Set**.*

Knowing this, at least we can make sense of the notion finitely generated free Θ-model: we have the diagram

$$\begin{tikzcd}
\text{f.g.}\Theta\text{models} \arrow[r] & \boldsymbol{Set}_T \arrow[r] & \Theta\text{-}\boldsymbol{Mod} \arrow[r, "\sim"] \arrow[d, shift left=2] \arrow[d, phantom, "\dashv"] & \boldsymbol{Set}^T \arrow[dl, leftrightarrow] \\
\boldsymbol{FinSet} \arrow[u] \arrow[rr] & & \boldsymbol{Set} \arrow[u, shift left=2] &
\end{tikzcd}$$

9.2.10 Proposition. *An algebraic theory Θ is canonically equivalent to the opposite of the category of finitely generated free Θ-models.*

9.2.11 Classical algebras as models. With the previous proposition, I think all the arguments from the discussion of groups go through in the general case. To start, given an algebra in the classical sense, we can define a model $\Theta \to \boldsymbol{Set}$ by

$$L \simeq \text{f.g. free algebras}^{\mathrm{op}} \to \text{algebras}^{\mathrm{op}} \xrightarrow{\mathrm{Hom}(-,G)} \boldsymbol{Set}$$

etc. . .

9.2.12 Theories and monads. We already saw (or claimed) that Θ-***Mod*** is monadic over ***Set***. The resulting monad is

$$\begin{array}{rcl} T : \textbf{\textit{Set}} & \longrightarrow & \textbf{\textit{Set}} \\ X & \longmapsto & \operatorname*{colim}_{n \subset X} F(\mathbf{n}) \end{array}$$

where $F(\mathbf{n})$ denotes the free Θ-model on n generators. This definition already shows that T is finitary. In fact:

9.2.13 Proposition. (Linton.) *Lawvere theories are essentially the same thing as finitary monads.*

9.2.14 Example. In the theory of commutative semirings, the set of all m-ary operations is precisely the set $\mathbb{N}[X_1, \ldots, X_m]$ of all polynomials in m variables with natural-numbers coefficients. Indeed, each polynomial P can be interpreted as the m-ary operation that sends $(a_1, \ldots, a_m)$ to $P(a_1, \ldots, a_m)$, i.e. substituting the constants into the polynomial.

Proof of Tambara's theorem

9.2.15 Products in $\mathbb{T}$. The arrows in $\mathbb{T}$ can be interpreted by extension as polynomial functors between slices of the form ***Set***$/m$, and here we have the isomorphism

$$\textbf{\textit{Set}}/(m+n) \simeq \textbf{\textit{Set}}/m \times \textbf{\textit{Set}}/n.$$

It follows that $m+n$ is the product of m and n in $\mathbb{T}$. The projection $m+n \to m$ is given as the pullback along the sum inclusion in ***Set***. Namely if $i_1 : m \hookrightarrow m+n$ denotes the first sum inclusion in the category of sets, then $i_1^* : \textbf{\textit{Set}}/(m+n) \to \textbf{\textit{Set}}/m$ is the first projection.

9.2.16 Idea of proof. The point is that for the two ***Set***-maps

$$0 \xrightarrow{e} 1 \xleftarrow{m} 2$$

the polynomial functor

$$m_! : \quad \begin{tikzcd} & 2 \arrow[r, "="] \arrow[dl, "="'] & 2 \arrow[dr, "m"] & \\ 2 & & & 1 \end{tikzcd}$$

considered as a map in $\mathbb{T}$, represents addition,

$$m_* : \quad \begin{tikzcd} & 2 \arrow[r, "m"] \arrow[dl, "="'] & 1 \arrow[dr, "="] & \\ 2 & & & 1 \end{tikzcd}$$

represents multiplication,

$$e_! : \quad \begin{tikzcd} & 0 \arrow[r, "="] \arrow[dl, "="'] & 0 \arrow[dr, "e"] & \\ 0 & & & 1 \end{tikzcd}$$

represents the neutral for addition, and

$$e_* : \quad \begin{tikzcd} & 0 \arrow[r, "e"] \arrow[dl, "="'] & 1 \arrow[dr, "="] & \\ 0 & & & 1 \end{tikzcd}$$

represents the neutral for multiplication. These are the four standard generating operations

$$\begin{tikzcd} 0 \arrow[r, shift left, "\ulcorner 0 \urcorner"] \arrow[r, shift right, "\ulcorner 1 \urcorner"'] & 1 & 2 \arrow[l, shift right, "+"'] \arrow[l, shift left, "\times"] \end{tikzcd}$$

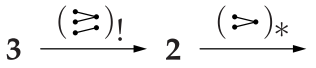
in the theory of commutative semirings. It is clear from these definitions that addition as well as multiplication satisfy the associative and unit laws.

We already saw that pullback provides the projection for the product in $\mathbb{T}$. It is also needed to account for distributivity, which in syntactic terms involves duplicating elements. Let us show how to derive use the abstract distributive law (8.3.5) to compute

$$m_* \circ k_!$$

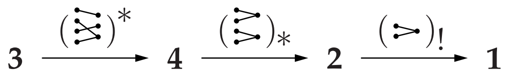
where $k : 3 \to 2$ is the map pictured as , recovering the distributive law $a(x+y) = ax + ay$ of elementary algebra.

The elementary distributive law means that



$$(a, x, y) \;\mapsto\; (a, x+y) \;\mapsto\; a(x+y)$$

is equal to



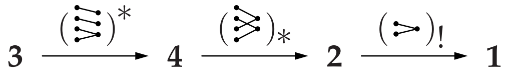
$$(a,x,y) \ \mapsto (a,x,a,y) \mapsto \ (ax,ay) \ \mapsto \ ax+ay$$

But showing this is precisely the pentagon-shaped distributivity diagram:

$$\begin{tikzcd}
 & \mathbf{4} \arrow[r, "\text{lowerstar}"] \arrow[dl, "\text{upperstar}"'] & \mathbf{2} \arrow[dd, "\text{lowershriek}"] \\
\mathbf{3} \arrow[dr, "\text{lowershriek}"'] & & \\
 & \mathbf{2} \arrow[r, "\text{lowerstar}"'] & \mathbf{1}
\end{tikzcd}$$

The set **2** appears as $f_*\mathbf{3}$, because the map k has one fibre of cardinality 2 and one of cardinality 1, and we have to multiply those two fibres. The trickiest is probably to compute the map $\varepsilon : \mathbf{4} \to \mathbf{3}$.

Note that there is an isomorphism that can be inserted in different places in the decomposition, depending on choices of constructing the 'distributivity diagram'. Above we had this isomorphism as part of the evaluation map ε. We could also have taken another pullback for **4**, then we would find the factorisation



$$(a,x,y) \ \mapsto (a,a,x,y) \mapsto \ (ax,ay) \ \mapsto \ ax+ay$$

9.2.17 Interpretation in terms of free semirings. In general we know that the Lawvere theory for P-algebras (for some abstract P), is equivalent to the opposite of the category of finitely generated free P-algebras. In our case this means that the category $\mathbb{T}$ should be equivalent to the opposite of the category of finitely generated free commutative semirings. Here is the equivalence in explicit terms.

The object $\mathbf{n}$ of $\mathbb{T}$ corresponds to the object $\mathbb{N}[x_1,\ldots,x_n]$ of the category of finitely generated free commutative semirings. A morphism $\mathbf{m} \to \mathbf{n}$ in $\mathbb{T}$ is given by

$$\mathbf{m} \leftarrow E \to B \to \mathbf{n}$$

which we think of as

$$\begin{array}{ccc} \boldsymbol{Set}/m & \longrightarrow & \boldsymbol{Set}/n \\ (X_i \mid i \in m) & \longmapsto & (\sum_{b \in B_j} \prod_{e \in E_b} X_{s(e)} \mid j \in n) \end{array}$$

so that's n polynomials P_i in m variables. To this map there should correspond a semiring homomorphism

$$F : \mathbb{N}[x_1, \ldots, x_n] \to \mathbb{N}[y_1, \ldots, y_m]$$

Thanks to the natural isomorphism $\mathbb{N}[x_1, \ldots, x_n] \simeq \mathbb{N}[x_1] \otimes \cdots \otimes \mathbb{N}[x_n]$, to give F is equivalent to giving for each $i = 1, 2, \ldots, n$ a semiring homomorphism

$$\mathbb{N}[x_i] \to \mathbb{N}[y_1, \ldots, y_m]$$

and such a semiring homomorphism is specified just by saying where x_i is sent. But we just send it to $P_i(y_1, \ldots, y_m)$. In other words, substitute the variables y_j into the formal expression of P_i. So the whole point is that finite polynomials can be interpreted in any semiring!

Conversely, given the semiring homomorphism F, we get in particular the images of the generators x_i, which we denote P_i. It's a polynomial in the y_i. We can interpret it as a finite polynomial functor $\boldsymbol{Set}/m \to \boldsymbol{Set}$, and hence an arrow in $\mathbb{T}$.

9.3 Differential calculus of polynomial functors

Introduction

Partial derivatives

In two variables, given $\{1,2\} \leftarrow E_1 + E_2 \to B$, the derivative with respect to 1 is

$$\{1,2\} \leftarrow (E_1 \times_B E_1) \smallsetminus \Delta \;+\; E_1 \times E_2 \to E_1$$

where the summand $(E_1 \times_B E_1) \smallsetminus \Delta$ maps to 1 and the summand $E_1 \times E_2$ maps to 2.

Spell it out:

$$\begin{aligned}
(X_1, X_2) \quad \longrightarrow \quad & \sum_{e} \prod_{u \in ((E_1 \times_B E_1) \smallsetminus \Delta + E_1 \times E_2)_e} X_{s(u)} \\
= \quad & \sum_{e} \prod_{u \in ((E_1 \times_B E_1) \smallsetminus \Delta)_e} X_1 \prod_{u \in E_1 \times E_2)_e} X_2 \\
= \quad & \sum_{e \in E_1} X_1^{E_1 - \{e\}} X_2^{E_2}
\end{aligned}$$

If we picture the fibre over $b \in B$ as a column of white dots (going to 1) and black dots (going to 2), then the new base is E_1, and the fibre over some white dot is the set of possible other dots in the same fibre. This means it can be another white dot or any black dot in that fibre, and this explains the shape of the formula.

So in short: the base is E_1, and for each $e \in E_1$ the new fibre is the complement of e in the old fibre.

9.3.1 Example. Now let us work out $\partial_1 \partial_2$. The result is

$$\begin{tikzcd}
E_1 \times_B E_1 \smallsetminus \Delta \times_B E_2 + E_1 \times_B (E_2 \times_B E_2 \smallsetminus \Delta) \arrow[d] \\
E_1 \times_B E_2
\end{tikzcd}$$

(and we should specify that the left-hand summand maps to 1 and the right-hand summand to 2). The conclusion is that the new base is the set of pairs (e_1, e_2), two dots in the same fibre and of different colour, and the fibre over such pair is the complement of that fibre.

WRITE OUT THE GENERAL FORMULA.

Homogeneous functors and Euler's Lemma

TIGHTEN UP THE FOLLOWING CHILDISH DISCUSSION

9.3.2 The degree of a homogeneous polynomial. The degree of the good-old monomial x^3y^2 is 5, and you compute that by 'forgetting' that x and y are distinct variables: considering them to be the same (say x), the monomial is $x^3x^2 = x^{3+2} = x^5$, and hence the degree is 5.

Similarly a monomial in many variables

$$\begin{tikzcd} & E \arrow[r] \arrow[dl, "s"'] & 1 \arrow[dr] & \\ I & & & 1 \end{tikzcd}$$

which is

$$(X_i \mid i \in I) \mapsto \prod_{e \in E} X_{s(e)}$$

Considering all the variable to be equal clearly this monomial is of degree E. In a sense we obtain this result by precomposing the functor with lowershriek along $d : I \to 1$ and then pullback to I . This yields $\prod_{e \in E} X_{d(s(e))} = \prod_{e \in E} X = X^E$.

More precisely, we actually started with a set over I then consider it a set over 1, this means lowershriek it along d: this destroys the difference between the variables. Now pull it back along first d then s.

This trick reduces the question to the one-variable case, and again we can define a homogeneous polynomial to be one

$$\begin{tikzcd} & E \arrow[r] \arrow[dl] & B \arrow[dr] & \\ I & & & J \end{tikzcd}$$

such that $E = B \times F$ for some set F.

Definition. A polynomial functor $I \leftarrow E \to B$ is *homogeneous of degree F* if $E = F \times B$. Here F is a fixed set. More generally, we will say that $I \leftarrow E \to B$ is homogeneous of degree F if there is specified an isomorphism (as polynomial functors) with one of the form $F \times B$.

Note that the map $s : F \times B \to I$ in general is different on each copy of F.

9.3.3 Example. It is evident that every monomial $E \to 1$ is homogeneous (of degree E).

9.3.4 Derivatives of monomials. Note that with the above definition of homogeneous, the derivative of a monomial is not naturally homogeneous: (say in the one-variable case), if $E \to 1$ is a monomial, then the derivative is $E \times E \smallsetminus \Delta \to E$. There is no natural way to trivialise this: there is no canonical way to provide a bijection $E \times E \smallsetminus \Delta \simeq (E \smallsetminus \{e\}) \times E$, as we remarked already in the definition of the derivative. In contrast, the differential operator XD provides a homogeneous polynomial: XD applied to $E \to 1$ gives $E \times E \to E$. And more generally, given a homogeneous polynomial $F \times B \to B$ then if we apply XD we get $F \times F \times B \to F \times B$, which is again homogeneous.

9.3.5 Homogenisation. Don't know what use of this. . . A homogenisation of $I \leftarrow E \to B$ (to degree F) consists in introducing a new variable and a diagram

$$\begin{tikzcd}
I \arrow[d, phantom, "\cap"] & E \arrow[l] \arrow[r] \arrow[d, phantom, "\cap"] & B \arrow[d, phantom, "\parallel"] \\
I+1 & F \times B \arrow[l] \arrow[r] & B
\end{tikzcd}$$

such that the complement of E in $F \times B$ is mapped to the new variable.

Let Λ be a fixed set. We denote by $\cdot\Lambda$ the functor

$$\begin{array}{rcl} \mathbf{\mathit{Set}}/I & \longrightarrow & \mathbf{\mathit{Set}}/I \\ X/I & \longmapsto & (\Lambda \times X)/I. \end{array}$$

It is represented by the linear polynomial $I \leftarrow \Lambda \times I = \Lambda \times I \to I$.

9.3.6 Lemma. *Let P be a homogeneous polynomial functor of degree F:*

$$I \leftarrow F \times B \to B$$

and let Λ be any set. Then there is a canonical natural isomorphism filling the square

$$\begin{tikzcd}
\mathbf{\mathit{Set}}/I \arrow[r, "\Lambda\cdot"] \arrow[d, "P"'] & \mathbf{\mathit{Set}}/I \arrow[d, "P"] \\
\mathbf{\mathit{Set}} \arrow[r, "\Lambda^F\cdot"'] & \mathbf{\mathit{Set}}
\end{tikzcd}$$

Proof. We should prove this in formal Beck-Chevalley style, but here is a quick and dirty proof: The upper-right-hand way around gives

$$\begin{aligned} X/I \mapsto (\Lambda \times X)/I \quad &\longmapsto \quad \sum_{b\in B} \prod_{e\in (F\times B)_b} (\Lambda \times X)_{s(e)} \\ &= \quad \sum_{b\in B} \prod_{e\in (F\times B)_b} (\Lambda \times X_{s(e)}) \\ &= \quad \sum_{b\in B} \prod_{e\in (F\times B)_b} \Lambda \prod_{e\in (F\times B)_b} X_{s(e)} \\ &= \quad \sum_{b\in B} \Lambda^F \times \prod_{e\in (F\times B)_b} X_{s(e)} \\ &= \quad \Lambda^F \times \sum_{b\in B} \prod_{e\in (F\times B)_b} X_{s(e)} \end{aligned}$$

which is precisely what the other way around gives. The equality signs represent canonical isomorphism. □

9.3.7 The Euler operator. Recall that the operator $X_i \cdot \partial_i$ sends $I \leftarrow E \rightarrow B$ to $E_i \times_B E \rightarrow E_i$ (no diagonal removed). Indeed, we know that first the partial derivative gives us $E_i \times_B E_i \smallsetminus E_i \times_B E \rightarrow E_i$, and then the X_i-multiplication consists in adding an *i*-point in each fibre. This is precisely to fill in back the diagonal.

Define the *Euler operator* to be

$$\text{Euler} := \sum_{i\in I} X_i \cdot \partial_i$$

Note that if P is given by $I \leftarrow E \rightarrow B$ then Euler P is given by

$$I \leftarrow E \times_B E \rightarrow B.$$

In other words, take the derivative as if all the variables were the same. . .

9.3.8 Lemma. *(Euler's lemma) If P : $\textbf{\textit{Set}}/I \rightarrow \textbf{\textit{Set}}$ is homogeneous of degree F then there is a natural isomorphism*

$$\text{Euler}\, P = F \times P$$

Proof. The classical proof is to write down the homogeneity equation

$$P(\Lambda \times X) = \Lambda^F \times P(X)$$

and recall that it is natural in Λ. Hence we can take derivative with respect to Λ. Now we have to invoke the chain rule

DO THAT

and finally set $\Lambda = 1$ to arrive at the promised formula.

The second proof exploits direct manipulation with the representing diagrams: we already observed that the Euler operator on $I \leftarrow E \rightarrow B$ gives $I \leftarrow E \times_B E \rightarrow E$. In the special case where the polynomial is homogeneous of degree F, this gives $F \times F \times B \rightarrow F \times B$. But this is clearly the same as $F \times P$. □

Note that Euler's lemma for good-old polynomials is also a triviality: by linearity it is enough to prove for monomials: each $X_i \partial_i$ just has the effect of placing the exponent of that variable as a coefficient. When summing all these we get of course the total degree.

9.3.9 Interpretation of variables as constants In order to better understand partial derivation we should investigate how to interpret other variables as constants.

9.3.10 Example. Given $\{1, 2\} \leftarrow E_1 + E_2 \rightarrow 1$, which is just $X_1^{E_1} X_2^{E_2}$, we want to think of $X_2^{E_2}$ as a constant B. The new polynomial is

$$\{1\} \leftarrow E_1 \times B \rightarrow B$$

Indeed,

$$X_1 \mapsto \sum_{b \in B} X^{E_1} = B \times X^{E_1}$$

as we wanted. The new polynomial is a family of polynomials parametrised by X_2. There is really some internal hom adjunction going on that we should get straight:

$$\frac{\boldsymbol{Set}_{I_1} \times \boldsymbol{Set}_{I_2} \longrightarrow \boldsymbol{Set}}{\boldsymbol{Set}_{I_2} \longrightarrow \boldsymbol{Poly}^{\mathrm{c}}(\boldsymbol{Set}_{I_1}, \boldsymbol{Set})}$$

WHAT IS IT?

9.4 Classical combinatorics

9.4.1 Binomial sets. Let I be a set, and let $k \in \mathbb{N}$. By definition the set

$$\binom{I}{k}$$

is the set of all subsets of I of cardinality k.

We also put $\mathscr{P}(I) = \sum_{k\in\mathbb{N}} \binom{I}{k}$, the set of all finite subsets of I.

Finally we shall need pointed versions: let

$$\binom{I}{k'}$$

denote the set of all pointed finite subsets of I of cardinality k. And put

$$\mathscr{P}'(I) = \sum_{k'\in\mathbb{N}'} \binom{I}{k'}$$

where the sum runs over all iso-classes of pointed finite sets.

One could prove a binomial theorem in this style: suppose I is finite of cardinality n. Then we have a (non-canonical) isomorphism of polynomial functors

$$(X_1 + X_2)^I \simeq \sum_{k=0}^{n} \binom{I}{k} X_1^k X_2^{n-k}$$

But there are non-canonical identifications, like collecting terms. In fact, the binomials appear as coefficients, something we know is not canonical. . .

The honest expansion is

$$(X_1 + X_2)^I = \sum_{S_1+S_2=I} X_1^{S_1} X_2^{S_2},$$

where the sum is over all partitions of I into two parts. The binomial theorem arises when we collect terms: collect all the S_1 of the same size k. . .

By the way, here is how to derive the honest formula using the distributive law. The functor $(X_1 + X_2)^I$ is the composite of $(X_1, X_2) \mapsto X_1 + X_2$ which is given by

$$2 \leftarrow 2 \rightarrow 2 \rightarrow 1$$

and the $Y \mapsto Y^I$ which is given by

$$1 \leftarrow I \rightarrow 1 \rightarrow 1.$$

To apply the distributive law, take first the pullback

$$\begin{tikzcd}
2 \times I \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & I \arrow[d] \\
2 \arrow[r] & 1
\end{tikzcd}$$

then we are dealing with

$$2 \times I \rightarrow I \rightarrow 1$$

and the distributive law says that

$$\prod_{b \in I} \sum_{c \in (2 \times I)_b} X_c = \sum_{m : I \to 2} \prod_{b \in I} X_{m(b)}$$

and the last sum can be written as the sum over all 2-part partitions,

$$= \sum_{S_1 + S_2 = I} X_1^{S_1} X_2^{S_2}.$$

We shall now make a similar argument for the elementary symmetric functions.

9.4.2 Elementary symmetric functions. By definition, the degree-k *elementary symmetric polynomial* of I-many variables is given by

$$I \leftarrow \binom{I}{k'} \rightarrow \binom{I}{k} \rightarrow 1$$

Since there is one functor for each $k \in \mathbb{N}$, we may as well put them all together in a single polynomial functor

$$I \xleftarrow{s} \mathscr{P}'(I) \xrightarrow{p} \mathscr{P}(I) \xrightarrow{t} \mathbb{N}$$

Here s returns the marked element of the subset, p forgets the mark on the subset, and t returns the cardinality of the subset.

As an example, if $I = \{1, 2, 3\}$ then the total symmetric function on I is the sequence

$$(1, X_1 + X_2 + X_3, X_1X_2 + X_1X_3 + X_2X_3, X_1X_2X_3, 0, \dots)$$

It is zero after coordinate 3, because there are no bigger subsets of I than of cardinality 3.

9.4.3 Symmetric functors. A (polynomial) functor $F : \textbf{\textit{Set}}/I \to \textbf{\textit{Set}}$ is called *symmetric* if it is invariant under permutation of the I-many variables. More precisely, for any permutation $\sigma : I \xrightarrow{\sim} I$, there is induced a polynomial functor $\sigma^* : \textbf{\textit{Set}}/I \xrightarrow{\sim} \textbf{\textit{Set}}/I$. We require that

$$F \circ \sigma^* \simeq F.$$

It should be noted that $\sigma^*(X_j \mid j \in I) = (X_{\sigma(i)} \mid i \in I)$. The permutation of coordinates could also be achieved using lowershriek or lowerstar. In that case the formulae are: $\sigma_!(X_i \mid i \in I) = (X_{\sigma^{-1}(j)} \mid j \in I)$ and $\sigma_*(X_i \mid i \in I) = (X_{\sigma^{-1}(j)} \mid j \in I)$. (The two functors coincide for isomorphisms.)

EXERCISE: the elementary symmetric functors are indeed symmetric in this sense.

Theorem (conjecture at the moment): *Every symmetric polynomial functor P factors through the total elementary symmetric functor.*

WARNING: the way symmetric polynomials factor through the elementary ones involves signs! Example:

$$xy^2 + xz^2 + yx^2 + yz^2 + zx^2 + zy^2$$

is clearly symmetric. To write it as a polynomial of elementary symmetric functions, the solution is

$$= e_1e_2 - 3e_3(x+y+z)(xy+xz+yz) - 3xyz$$

So in order to get a polynomial-functor version of this result, we need to take care of signs in an appropriate way.

More precisely, given any symmetric polynomial functor P:

$$I \leftarrow E \to B \to 1$$

(which we can assume to be furthermore homogeneous of degree d) there exists a polynomial functor

$$\{0,1,2,\ldots,d\} \leftarrow F \to C \to 1$$

such that the composite

$$I \leftarrow \mathscr{P}'(I) \to \mathscr{P}(I) \to \mathbb{N} \leftarrow F \to C \to 1$$

is isomorphic to P

If the proof is going to follow Lang [70], then we need the binomial theorem to prove first that the elementary symmetric functors arise as coefficients of

$$\prod_{i\in I}(Y + X_i),$$

but perhaps there is a more direct proof. . .

9.5 Polynomial functors on collections and operads

We saw in Chapter 5 that there is a natural equivalence of categories

$$\mathbf{Poly}^{\mathrm{c}}/M \simeq \mathbf{Set}/\mathbb{N}$$

the second category is the category of collections. In this section we take a look at a few constructions with polynomial functors from the viewpoint of collections: since $\mathbf{Set}/\mathbb{N}$ is a slice of $\mathbf{Set}$ it makes sense to study polynomial functors on it!

LOTS OF THINGS TO THROW AWAY. . .

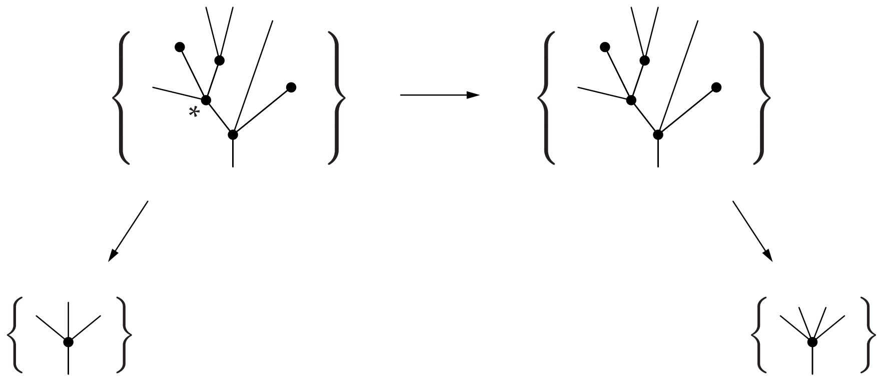
The free-operad functor

9.5.1 The free-operad functor. The free-operad functor (5.5.1),

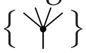
$$\begin{array}{ccc} \mathbf{Set}/\mathbb{N} & \longrightarrow & \mathbf{Set}/\mathbb{N} \\ A & \longmapsto & T(A) \end{array}$$

was defined in 5.5.1 as generated by the free-forgetful adjunction between collections and operads. It sends a collection A to the operad freely generated by it. It is the set of planar trees with nodes decorated in A subject to a compatibility condition: nodes with n input edges must have decoration in A_n.

Now it is easy to see that it is just the polynomial functor given by



(9.2)

(In this diagram as well as in the following diagrams of the same type, a symbol  is meant to designate the set of *all* bouquets like this (with the appropriate decoration), but at the same time the specific figures representing each set are chosen in such a way that they match under the structure maps.)

Now that we know it is a polynomial functor $T : \textbf{\textit{Set}}/\mathbb{N} \to \textbf{\textit{Set}}/\mathbb{N}$, we can look at the category of T-algebras (in the Lambek sense). It consists of collections $X/\mathbb{N}$ equipped with a map of collections $T(X) \to X$.

Let $C : \textbf{\textit{Set}}/\mathbb{N} \to T\textbf{\textit{alg}}$ denote the free T-collection functor. We are interested in the value of C at the terminal collection $\mathbb{N} \to \mathbb{N}$ which we denote by 1: it is the least fixpoint for the functor

$$X \mapsto \mathbb{N} + T(X)$$

This is the set of all planar constellations cf. 5.5.2

Since T is a monad, there is a map of collections $C(1) \to T(1)$.

Linear differential operators are linear

in the sense of polynomial functors

THERE IS A SERIOUS ISSUE TO SORT OUT HERE: WE ARE WORKING IN $\textbf{\textit{Set}}/\mathbb{N} \simeq \textbf{\textit{Poly}}^{\mathrm{c}}/M$. IS DERIVATION DEFINED AS A FUNCTOR ON THIS CATEGORY?

Yes. It depends on a trick: we know that the derivative of $P \to M$ is $P' \to M'$. But there exists a cartesian natural transformation $M' \to M$. It is given by

$$\begin{tikzcd}
\mathbb{N}'' \arrow[r, "\varphi"] \arrow[d, "u'"'] & \mathbb{N}' \arrow[d, "u"] \\
\mathbb{N}' \arrow[r, "n \mapsto n-1"'] & \mathbb{N}
\end{tikzcd}$$

where u denotes the usual universal family for finite sets, u' is its derivative, and φ is defined as

$$(i \neq j < n) \mapsto \begin{cases} j-1 < n-1 & \text{for } j > i \\ j < n-1 & \text{for } j < i \end{cases}$$

We really need to explore this map. I think it will play a central role.

The point is related to this: given $E \to 1$, usually when we take derivative of X^E we get $\sum_{e \in E} X^{E-\{e\}}$ and we do not make the noncanonical reduction to anything of the sort $E \times X^{E-1}$. There is no canonical way of identifying those punctured versions of E with any fixed set with one element less.

But in the special situation where we are over M, then every fibre acquires an order, and when we remove an element this order is retained, so using this order there is a canonical way of identifying the complement with something. Hence the case of polynomials over M is rather special. This is what we are exploiting.

A LOT OF JUNK TO THROW AWAY IN THIS PART

Given a finite map $p : E \to B$, there is an associated collection $B \to \mathbb{N}$ (the classifying map). We have studied the differentiation of p. It is the finite map

$$p' : E \times_B E \smallsetminus \Delta \to E$$

In terms of the polynomial functor it is

$$X \mapsto \sum_{e \in E} X^{E_{p(e)} - e}.$$

This is a finite map over E. To find the corresponding collection, we need to describe the degree map (the classifying map) $E \to \mathbb{N}$. The fibre over $e \in E$ has degree one less than the fibre $E_{p(e)}$. Hence e has degree n if and only if $p(e)$ has degree $n+1$. So the corresponding collection is just

$$\sum_{n \geq 0} (n+1) B_{n+1} \;\to\; \mathbb{N}$$

(Of course this is also what we expect to find in view of our experience with good old-fashioned polynomials.)

Now differentiation defines a functor from ***FinMap*** to itself. At least this is true if we restrict to cartesian squares as arrows in the category. We should check functoriality more carefully. We claim that there is correspondingly a true morphism of collections:

$$\textbf{\textit{Set}}/\mathbb{N} \to \textbf{\textit{Set}}/\mathbb{N}$$

(this is a functor).

9.5.2 Proposition. *The functor*

$$\begin{array}{rcl} \textbf{\textit{Set}}/\mathbb{N} & \longrightarrow & \textbf{\textit{Set}}/\mathbb{N} \\ \sum_{n \geq 0} A_n & \longmapsto & \sum_{n \geq 0} (n+1) A_{n+1} \end{array}$$

induced by differentiation of finite maps (polynomial functors) is itself polynomial—in fact it is linear.

Note that the fibre over $n = 0$ is thrown away.

It is the map

$$\begin{tikzcd}
& \mathbb{N} + \mathbb{N}' \arrow[r, "\text{id}"] \arrow[dl, "+1"'] & \mathbb{N} + \mathbb{N}' \arrow[dr] & \\
\mathbb{N} & & & \mathbb{N}
\end{tikzcd}$$

Here the right-hand map is the one corresponding to the polynomial $X \times M'(X)$. The left-hand map $\mathbb{N} + \mathbb{N}' \to \mathbb{N}$ is the one sending an element to one more than expected. Perhaps it is best to realise it in two steps:

$$\mathbb{N} + \mathbb{N}' \to \mathbb{N} \xrightarrow{+1} \mathbb{N}$$

Proof. Start with the collection $\sum_{n\geq 0} A_n \ \rightarrow\ \mathbb{N}$. Pulling back along $\mathbb{N} \stackrel{+1}{\rightarrow} \mathbb{N}$ gives the collection $\sum_{n\geq 0} A_{n+1} \ \rightarrow\ \mathbb{N}$. Now pull back to the set $\mathbb{N}+\mathbb{N}'$. The fibre over an n in the first summand is A_{n+1}. The fibre over a point $(a < b)$ in $\mathbb{N}'$, is A_{b+1} (and there are b fibres of this type). Finally, take lowershriek (sum along the canonical map $\mathbb{N}+\mathbb{N}' \rightarrow \mathbb{N}$. For each n in the target $\mathbb{N}$ there are $n+1$ elements in the fibre, and all these element has the same sort of fibre, namely A_{n+1}. So altogether, the resulting collection is $\sum_{n\geq 0}(n+1)A_{n+1}$. □

Note that there is another representation of the functor as a finite map:

$$\begin{tikzcd}
& \mathbb{N}' \arrow[r, "\mathrm{id}"] \arrow[dl] & \mathbb{N}' \arrow[dr, "-1"] & \\
\mathbb{N} & & & \mathbb{N}
\end{tikzcd}$$

The minus-one map makes sense in this situation because the map $\mathbb{N}' \rightarrow \mathbb{N}$ factors through $\mathbb{N} \smallsetminus \{0\}$, from which is minus-one map is well-defined. So the precise definition of the map is

$$\mathbb{N}' \rightarrow \mathbb{N} \smallsetminus \{0\} \xrightarrow{-1} \mathbb{N}$$

9.5.3 Remark. The second representation is better, I think. Start with the collection $\sum_{n\geq 0} A_n \ \rightarrow\ \mathbb{N}$. Pull it back to $\mathbb{N}'$: the fibre over a point $(i < n)$ is just A_n. Finally sum, by going down to $\mathbb{N} \smallsetminus \{0\}$: the fibre over $n \geq 1$ is now nA_n. Finally, go down along the minus-one shifting map: the fibre over n will be $(n+1)A_{n+1}$. That's all.

Note that if we compose with the plus-one map, then we undo the last shift: we get in the end nA_n. This is the operator xD. Se below.

9.5.4 Remark. Some related possibilities and their result:

Omitting the plus-one map on the left, we get the operator sending $\sum_{n\geq 0} A_n$ to $\sum_{n\geq 0}(n+1)A_n$. Indeed, the fibres never change in this construction, only the coefficients.

Using the plus-one map on the left, but using the standard $\mathbb{N}'$ in the middle (instead of $\mathbb{N}+\mathbb{N}'$) the result will be the operator sending $\sum_{n\geq 0} A_n$ to $\sum_{n\geq 0} nA_{n+1}$. Indeed, in this case the number of fibres of degree n is still just n...

9.5.5 Remark. Note that the pushforth along some map in the middle will perform multiplication of some sort (i.e. give something like $\prod V_e$. So such polynomial functors are of no use to realise derivative-like operations.

For example, pushforth along $\mathbb{N} \stackrel{+1}{\to} \mathbb{N}$ with create a constant terms 1 for the empty fibre! Indeed, there will be an empty product, and hence a term 1. This is true in general: each empty fibre will produce a constant term 1.

9.5.6 Remark. Note that the single operation: pullback along $\mathbb{N} \stackrel{+1}{\to} \mathbb{N}$ gives the operator

$$\sum_{n\geq 0} A_n \mapsto \sum_{n\geq 0} A_{n+1}$$

This would be some sort of divided-power derivation. I don't know anything about such operators if they have any use.

9.5.7 Example. The poly-map diagram whose last part is lowershrieking along $\mathbb{N} \stackrel{+1}{\to} \mathbb{N}$ gives rise to the functor

$$\begin{array}{ccc} \textbf{\textit{Set}}/\mathbb{N} & \longrightarrow & \textbf{\textit{Set}}/\mathbb{N} \\ \displaystyle\sum_{n\geq 0} A_n & \longmapsto & \displaystyle\sum_{n\geq 1} A_{n-1} \end{array}$$

This is the operator multiplication with X.

Now it follows that the operator xD is also polynomial: first the derivative polynomial, then lowershriek along the plus-one map. Note that there is now a plus-one map in each end. In fact the plus-one maps out of $\mathbb{N} + \mathbb{N}'$ amount just to undo the effect of having an extra summand. In fact, the operator xD can be described directly, and much more easily by

$$\begin{tikzcd} & \mathbb{N}' \arrow[r, "\mathrm{id}"] \arrow[dl] & \mathbb{N}' \arrow[dr] & \\ \mathbb{N} & & & \mathbb{N} \end{tikzcd}$$

9.6 Bell polynomials

References:

–Faà di Bruno formula and Hopf algebra

As a starting point, see Figueroa and Gracia-Bondía [34], Part II.

See also: Knuth [61].

See also: Touchard polynomials (Wikipedia, printed)

Let S be a set. A k-partition of S consists of k nonempty subsets called *blocks*, which are pairwise disjoint and whose union is S. (We only care about which elements of S are together in a block, not about the order of the blocks.)

Let $B_{n,k}$ denote the number of k-partitions of an n-element set. (Note that 0-partitions do not exist for $n > 0$. But the empty set can well be partitioned into 0 (non-empty!) parts.) Let $B_n := \sum_{k=0}^{n} B_{n,k}$ denote the number of partitions of an n-element set. These are called the Bell numbers. (Note that according to our convention we have $B_{0,0} = 1$.)

More generally, we define the Bell polynomials, $B_n(x_1, \ldots, x_n)$ a polynomial in n ordered variables. Again we have its homogeneous terms, so that $B_n(x_1, \ldots, x_n) = \sum_{k=0}^{n} B_{n,k}(x_1, \ldots, x_n)$. The idea is that $B_{n,k}(1, \ldots, 1) = B_{n,k}$. The variables are meant to keep track of the shape of the involved partitions: instead of just saying that there are $b_{4,2} = 7$ 2-partitions of the 5-element set, we now write $B_{4,2}(x_1, \ldots, x_4) = 4x_1x_3 + 3x_2^2$ meaning that the partitions naturally divide into 4 partitions of shape $1 + 3$ and 3 partitions of shape $2 + 2$. So the variables are to keep track of the shapes, and the coefficients count how many there are of a given shape. Let λ denote a shape of a k-partition of an n-element set. We are thus defining

$$B_{n,k}(x_1, \ldots, x_n) = \sum_{\lambda} b_\lambda x_\lambda$$

Here b_λ denotes the number of (n,k)-partitions of shape λ, and x_λ denotes the monomial obtained from the partition shape.

It is fruitful to consider all the Bell polynomials as polynomials is countably many variables, $x_0, x_1, x_2, \ldots$. Then we just observe that the 0th Bell polynomial is $B_0(\mathbf{x}) = x_0$, and that the polynomial $B_n(\mathbf{x})$ only depends on the n variables $x_1, \ldots, x_n$.

We can categorify this construction: let S be an n-element set, and consider also the nth ordinal $[n] = \{1, 2, \ldots, n\}$ — with no further relationship between S and $[n]$. Define a polynomial functor B_n by the diagram

$$\begin{tikzcd}
 & \Pi'(S) \arrow[rr] \arrow[dl] & & \Pi(S) \arrow[dr] & \\
{[n]} & & B_n & & {[1]}
\end{tikzcd}$$

Here $\Pi(S)$ denotes the set of partitions of S, and $\Pi'(S)$ denotes the set of partitions with one marked block. The map $\Pi'(S) \to [n]$ returns the size of the marked partition.

Spelling out what the functor does, it sends an $[n]$-indexed set X (i.e. an ordered set of variable $X_1, \ldots, X_n$) to the set given by: it has one monomial for each element in $\Pi(S)$. Each monomial is of degree k if the partition is a k-partition (because the instruction is 'multiply along the fibres'), and the precise monomial we find here is obtained by multiplying the sizes of the k blocks: if there is a block of size i, put a factor X_i. It is clear that this is precisely the polynomial functor categorifying the informal description we first gave.

Here is an example. Consider the 4-element set $\{a, b, c, d\}$. The possible partitions are:

$$\begin{array}{ll}
k = 1: & (abcd) \\
k = 2: & (abc)(d) \\
 & (abd)(c) \\
 & (acd)(b) \\
 & (bcd)(a) \\
 & (ab)(cd) \\
 & (ac)(bd) \\
 & (ad)(bc) \\
k = 3: & (ab)(c)(d) \\
 & (ac)(b)(d) \\
 & (ad)(b)(c) \\
 & (bc)(a)(d) \\
 & (bd)(a)(c) \\
 & (cd)(a)(b) \\
k = 4: & (a)(b)(c)(d)
\end{array}$$

So the polynomial functor is

$$(X_1, X_2, X_3, X_4) \longmapsto \underbrace{X_4}_{k=1} + \underbrace{4X_1X_3 + 3X_2^2}_{k=2} + \underbrace{6X_1^2X_2}_{k=3} + \underbrace{X_1^4}_{k=4}$$

Note that since the indices on the variable count something, Bell polynomial do not give meaning on abstract variable sets, only when the indexing set is an ordinal.

Each Bell polynomial B_n is a functor of n variables. We might as well say that it is a functor of countably may variables, then we have them all on the same footing — it doesn't matter that in a given polynomial only some of the variables are ever used. So now the nth Bell polynomial reads

$$\begin{tikzcd}
& \Pi'(S) \arrow[rr] \arrow[dl] & & \Pi(S) \arrow[dr] & \\
\mathbb{N} & & B_n & & 1
\end{tikzcd}$$

Now let us assemble all the Bell polynomials into a single one: this means a vector of polynomials, where we put the nth polynomial as the nth coordinate functor. So the total Bell polynomial is

$$\begin{tikzcd}
& \sum_{n\in\mathbb{N}} \Pi(n)' \arrow[rr, "p"] \arrow[dl, "s"'] & & \sum_{n\in\mathbb{N}} \Pi(n) \arrow[dr, "t"] & \\
\mathbb{N} & & B & & \mathbb{N}
\end{tikzcd}$$

Here the t-fibre is the set of all partitions of some standard n-element set $S = [n]$.

But now we may as well forget about standard n-element sets, and at the same time resolve the mysterious size function in the definition of s: we now define the total Bell polynomial to be

$$\begin{tikzcd}
& \text{partitions}' \arrow[rr, "p"] \arrow[dl, "s"'] & & \text{partitions} \arrow[dr, "t"] & \\
\text{FinSet} & & B & & \text{FinSet}
\end{tikzcd}$$

This is much more canonical. Note however that it is a little bit different from the original, because since subsets of a given set are different sets even though they may have the same size, it implies that there will be no coefficients in this new version. For example, in the polynomial B_4, which we really ought to write as $B_{\{a,b,c,d\}}$ — it is the $\{a,b,c,d\}$-component of B — instead of having as before a term $6X_1^2X_2$, we will now have six separate terms. Namely, the variables are no longer indexed by natural numbers, but rather by actual sets, namely the subsets in question. So the term expands to

$$X_{\{ab\}}X_{\{c\}}X_{\{d\}} + X_{\{ac\}}X_{\{b\}}X_{\{d\}} + X_{\{ad\}}X_{\{b\}}X_{\{c\}} + \\ X_{\{bc\}}X_{\{a\}}X_{\{d\}} + X_{\{bd\}}X_{\{a\}}X_{\{c\}} + X_{\{cd\}}X_{\{a\}}X_{\{b\}}$$

Now the polynomial functor has become something we can easily grasp, and it looks familiar: in fact we claim it is a monad. It even looks suspiciously like some free monad, or at least the level-2 part: think of the finite sets as elementary trees, and think of the partitions as refinements where each leaf has been refined to another elementary tree, i.e. we have grafted other bouquets on top.

Better still: the types of this polynomial functor are the finite sets. The operations are shrubs such that each input edge is decorated by a finite set, and the output edge is decorated by the disjoint union of those sets. Beware that we should divide out by some symmetry group, because if a shrub has arity 2, and we have two given sets, there are two ways to decorate that particular shrub with those two sets, but we want that to count as only one operation!

In other words, the node is the operation 'take disjoint union'. Now if we apply this endofunctor to itself, the set of operations will be the set of 2-level trees whose edges are decorated by finite sets, subject to the compatibility condition that the outgoing edge of a node is decorated by the sum of the sets decorating the incoming edges of that node.

[added 2011-07-10] We need groupoids to do this properly. So consider the groupoid of finite sets and bijections, the category of partititions, and the category of partititions with a marked block. That's the total Bell polynomial.

Now as a more convenient model for partitions, we consider surjections. An isomorphisms between two surjections $E \twoheadrightarrow B$ and $E' \twoheadrightarrow B'$ is a pair of isomorphisms, $E \simeq E'$ and $B \simeq B'$ forming a commutative square.

The category of surjections is equivalent to that of partitions. It's advantage is that it has some concrete sets involves.

$$\begin{tikzcd}
& \text{pointed surjections} \arrow[rr, "p"] \arrow[dl, "s"'] & & \text{surjections} \arrow[dr, "t"] & \\
\mathbb{B} & & B & & \mathbb{B}
\end{tikzcd}$$

The map t sends a surjection $E \twoheadrightarrow B$ to the total space E. The groupoid of pointed surjections has as objects diagrams

$$\begin{tikzcd}
& E \arrow[d] \\
1 \arrow[r, "b"'] & B
\end{tikzcd}$$

and the map p just forgets the pointing. The map s sends such a pointed surjection to the fibre E_b.

Now, when figuring out what this big polynomial does, we can work component-wise. So suppose we are only interested in the E-component of the total polynomial. We proceed as before. Note that there is a variable for every finite set!, and the particular component we are looking at only picks out very few fibres! Here is the input: its a groupoid X with a functor to $\mathbb{B}$. In fact, we would like that fibres over isomorphic finite sets should be isomorphic. Probably what we want is more precisely that it should be a fibration. In other words, the input to the polynomial is a species! This means that the input family is essentially determined by its fibres over some standard finite sets $\mathbf{n}$. Similarly, since all involved constructions are functorial, the E-component will be a groupoid equivalent to the E' component, if $E \simeq E'$. And in a more invariant manner, the E-component will not just be a six-element sum of products of some strict fibres (corresponding to the subsets of E), it will rather be a huge groupoid equivalent to this. The π_0 of this groupoid should be like the discrete case we considered first. But furthermore, some of these components will be equivalent: for example $X_{\{ab\}}X_{\{c\}}X_{\{d\}} \simeq X_{\{ac\}}X_{\{b\}}X_{\{d\}}$, so that we get six equivalent terms in this example.

The value of this total polynomial on the terminal species is as always the upper-right-hand corner of the bridge diagram, in this case the groupoid of surjections. More generally, the value of the total polynomial on an arbitrary species is the groupoid of all ways a partition can be decorated by that species. What does this mean concretely? It seems to mean: partitions where each part is equiped with an X-structure.

Let $\mathbb{S}$ denote the groupoid of partitions (or more precisely, surjections).

Here is another polynomial functor:

$$\begin{tikzcd}
 & \mathbb{S} \arrow[r, "\mathrm{id}"] \arrow[dl, swap, "r"] & \mathbb{S} \arrow[dr, "t"] & \\
\mathbb{B} & & & \mathbb{B}
\end{tikzcd}$$

If applied to a species X, the result is the groupoid of partitions, with an X structure on the set of parts.

Now combine these two polynomials:

$$\begin{tikzcd}
 & \mathbb{S}' + \mathbb{S} \arrow[rr, "\langle p{,}\mathrm{id}\rangle"] \arrow[dl, swap, "s+r"] & & \mathbb{S} \arrow[dr, "t"] & \\
\mathbb{B} + \mathbb{B} & & \circ & & \mathbb{B}
\end{tikzcd}$$

This is the polynomial functor effectuating composition of species! (I think — more verification is required.)

Namely, given the value of the polynomial on a pair of species (G, F) (that's the same as groupoid over $\mathbb{B} + \mathbb{B}$), is the groupoid of partitions with a G-decoration on each part and an F-decoration on the set of parts.

Observing that the algebra of partitions is just a crude version of the algebra of trees, this result is analogous to my categorification of the Hopf algebra of trees.

CERTAINLY A LOT OF STUFF TO EXERCISE HERE

Chapter 10

Categories and bicategories of polynomial functors

ch:bicat

Many results in this chapter and the next are due to joint work with Nicola Gambino [39].

10.1 Natural transformations between polynomial functors

In this section we mostly keep two sets I and J fixed. Let $\boldsymbol{PolyFun}(I,J)$ denote the full subcategory of $\boldsymbol{Cat}(\boldsymbol{Set}/I,\boldsymbol{Set}/J)$ consisting of the polynomial functors (i.e. those isomorphic to one of the form of our diagrams), and all natural transformations between them.

10.1.1 Morphisms. We argued in Chapter 2 that in the one-variable case cartesian squares and backwards triangles give rise to natural transformations between polynomial functors, and that all natural transformations arise as composition of a backwards triangle followed by a cartesian square. The same is true in the many-variable case, and in fact the proofs will now seem much easier, because we have the more abstract viewpoints of adjunctions and Beck-Chevalley conditions at our disposal.

10.1.2 Representable morphisms. Given a diagram

$$\begin{tikzcd}
& E \arrow[r, "p"] \arrow[dl, "s"'] & B \arrow[dr, "t"] \arrow[dd, equal] & \\
I & & & J \\
& F \arrow[uu, "w"'] \arrow[ul, "z"] \arrow[r, "q"'] & B \arrow[ur, "t"'] &
\end{tikzcd}$$

then there is induced a natural transformation

$$t_! \, p_* s^* \overset{\eta}{\Rightarrow} t_! \, p_* w_* w^* s^* = t_! \, q_* z^*$$

where $\eta : 1 \Rightarrow w_* w^*$ denotes the unit for the adjointness $w^* \dashv w_*$.

[One might think that conversely every natural transformation between these polynomial functors would be induced in this way, but this is not true: for example the polynomial functor $1 \leftarrow 2 \rightarrow 2 \rightarrow 1$ (which sends X to $X + X$) has a nontrivial automorphism given by the twist map (interchange of factors). This one is not induced by a diagram like the above. What is true is that every natural transformation for which the terminal-object component is the identity on B does come like this, cf. the Yoneda lemma below.]

[Here there is a problem that we want to describe natural transformations $t_! \, p_* s^* \overset{\eta}{\Rightarrow} t_! \, p_* w_* w^* s^* = t_! \, q_* z^*$ by cancelling away $t_!$. In order to do this we would like $\mathrm{Nat}(P, Q) \to \mathrm{Nat}(t_! \, P, t_! \, Q)$ to be a bijection or at least a surjection. This is equivalent to requiring the functor

$$\begin{array}{rcl}
\mathbf{Fun}(\mathbf{Set}/I, \mathbf{Set}/B) & \longrightarrow & \mathbf{Fun}(\mathbf{Set}/I, \mathbf{Set}/J) \\
P & \longmapsto & t_! \, P
\end{array}$$

to be fully faithful or at least full. This is not the case in general though: for example if $J = 1$, then it is not true that given X/B and Y/B then every arrow $X \to Y$ over I is also over B. It is true that $t_!$ is faithful, but this doesn't help. What really does the job is the fact that all components of the natural transformation lie over $P(1) \to Q(1) \ldots$]

10.1.3 Cartesian morphisms. Given a diagram

$$\begin{tikzcd}
& \overline{E} \arrow[r, "\overline{p}"] \arrow[dl, "\overline{s}"'] \arrow[dd, "u"] \arrow[ddr, phantom, "\lrcorner", very near start] & \overline{B} \arrow[dr, "\overline{t}"] \arrow[dd, "k"'] & \\
I & & & J \\
& E \arrow[ul, "s"] \arrow[r, "p"'] & B \arrow[ur, "t"'] &
\end{tikzcd}$$

then there is induced a natural transformation

$$\overline{t}_! \, \overline{p}_* \overline{s}^* = t_! \, k_! \, \overline{p}_* u^* s^* \overset{\text{B-C}}{=} t_! \, k_! \, k^* p_* s^* \Rightarrow t_! \, p_* s^*$$

where the last step was the counit for the adjunction $k_! \dashv k^*$. Now the counit for this adjunction is always cartesian, cf. Lemma 8.2.10. Whiskering with $t_!$ and $p_* s^*$ do not alter the cartesianness, because each of these functors preserve pullbacks, cf. 1.7.1.

We can now prove a basic theorem about polynomial functors:

esentation-thm

10.1.4 Theorem. *Every natural transformation between polynomial functors factors uniquely as a representable natural transformation followed by a cartesian one.*

Proof. The statement is: given a diagram

$$\begin{tikzcd}
& E \arrow[r, "p"] \arrow[dl, "s"'] & B \arrow[dr, "t"] & \\
I & & & J \\
& F \arrow[ul, "z"] \arrow[r, "q"'] & C \arrow[ur, "m"'] &
\end{tikzcd}$$

and a natural transformation $t_! \, p_* s^* \Rightarrow m_! \, q_* z^*$ then there exist maps

$$\begin{tikzcd}
& E \arrow[r, "p"] \arrow[dl, "s"'] & B \arrow[d, equal] \arrow[dr, "t"] & \\
I & F \times_C B \arrow[l, "g"] \arrow[u, "w"'] \arrow[r, "r"] \arrow[d, "v"] & B \arrow[r] \arrow[d, "u"] & J \\
& F \arrow[ul, "z"] \arrow[r, "q"'] & C \arrow[ur, "m"'] &
\end{tikzcd}$$

inducing the given natural transformation. The uniqueness statement is clarified by this: if this diagram exists it is unique up to unique isomorphism by the universal property of the pullback.

The key point is that the terminal-object component of the natural transformation gives us a J-map $u : B \to C$. Since all other components lie over this one, they are all C-maps, so we reduce to the task of describing all natural transformations $u_! \, p_* s^* \Rightarrow q_* z^*$, and by adjointness these correspond to the natural transformations

$$p_* s^* \Rightarrow u^* q_* z^*.$$

Now form the pullback square in the desired diagram, and apply Beck-Chevalley to it: we have reduced to a natural transformation

$$p_* s^* \Rightarrow r_* v^* z^* = r_* g^*$$

which by the following Yoneda lemma must be induced by a unique $w : F \to E$. It is clear from the construction of u and w that the diagram induces the original natural transformation. □

10.1.5 Yoneda lemma. *Given a diagram*

$$\begin{tikzcd}
& E \arrow[ld, "s"'] \arrow[rd, "p"] & \\
I & & B \\
& F \arrow[lu, "z"] \arrow[ru, "q"'] &
\end{tikzcd}$$

the canonical map

$$\begin{aligned}
\mathrm{Hom}_{I,B}(F,E) &\longrightarrow \mathrm{Nat}(p_* s^*, q_* z^*) \\
w &\longmapsto [p_* s^* \overset{\eta}{\Rightarrow} p_* w_* w^* s^* = q_* z^*]
\end{aligned}$$

is a bijection.

Proof. Since lowerstarring is a fibrewise construction we reduce to the case $B = 1$. In this case $p_* s^*$ is just the functor $X/I \mapsto \mathrm{Hom}_I(E/I, X/I)$, and the result follows from the usual Yoneda lemma for ***Set***$/I$.

Some details: $p_* s^* X = \prod_{e \in E}(X \times_I E)_e = \prod_{e \in E} X_{s(e)} = \mathrm{Hom}_I(E/I, X/I) = \prod_{i \in I} X_i^{E_i}$. □

thm:carttopoly

10.1.6 Lemma. *If P is a polynomial functor and $\phi : Q \Rightarrow P$ is a cartesian natural transformation, then Q is polynomial.*

Proof. Since $\phi : Q \Rightarrow P$ is cartesian, we have an isomorphism

$$Q(X) \cong P(X) \times_{P(1)} Q(1)$$

Suppose P is represented by $I \leftarrow E \rightarrow B \rightarrow J$, and recall that $P(1) \cong B$. We now define $C = Q(1)$, let $h : C \to B$ be the composite of $\phi_1 : Q(1) \to P(1)$ with the isomorphism $P(1) \cong B$, and put $F = E \times_B C$. It is now clear that the polynomial functor represented by $I \leftarrow F \rightarrow C \rightarrow J$ is isomorphic to Q, since both have a cartesian natural transformation to P and agree on 1. □

Theorem 10.1.4 says that there is a factorisation system on $\textbf{\textit{PolyFun}}(I,J)$, but note that the class of representable morphisms is not saturated in the sense that it does not contain all isomorphisms: it only contains isomorphisms $P \xrightarrow{\sim} Q$ for which $P(1) \to Q(1)$ is the identity map. These are precisely the vertical arrows for the functor $P \mapsto P(1)$, and the theorem can be formulated a little bit more precisely as follows:

10.1.7 Theorem. *The functor*

$$\begin{array}{rcl} \textbf{\textit{PolyFun}}(I,J) & \longrightarrow & \textbf{\textit{Set}}/J \\ P & \longmapsto & P(1) \end{array}$$

is a Grothendieck fibration. The cartesian arrows are precisely the cartesian natural transformations.

The importance is not just the ability to factor natural transformations, but as much the mere fact that every natural transformation can be captured on the level of the representing sets, so that all computations become mere manipulations of sets.

Furthermore the following important corollary results: Consider the category whose object are the diagrams representing polynomial functors. The arrows are given as the natural transformations of the induced functors. Then by construction, the inclusion is fully faithful, and the essential image, the category of all functors isomorphic to a polynomial one is an equivalent category.

Note that even if a given functor is isomorphic to a polynomial one, then there may be several different isomorphisms. This just amounts to saying that some polynomial functors admit automorphisms. For example $1 \leftarrow \mathbb{N}' \to \mathbb{N} \to 1$ has automorphisms which are even vertical for the fibration, given by permuting the elements of the fibres of $\mathbb{N}' \to \mathbb{N}$.

In fact we must absolute work out this example, seeing which natural transformation is induced by such an automorphism. In fact this example can be observed already in the very simple example

$$1 \leftarrow 2 \to 1 \to 1$$

with an automorphism of 2. It is the functor

$$X \mapsto X \times X$$

and surely the natural transformation induced by the automorphism is the twist. However, working out this example in terms of representable or cartesian natural transformation is quite tricky! As far as I remember, it is easy to see that the twist arises from the representable natural transformation induced by the automorphism, but that it is much trickier to see that also the cartesian natural transformation induced by the automorphism is the twist. . . Possibly this involves also the pseudo-naturality of the pullback!

Note that these calculations could actually be performed and explained already in the one-variable case, but I think it is more appropriate to do it in the many-variable case. . .

Perhaps there is a separate argument to be seen in the internal language, which would then be a suitable argument for the one-variable viewpoint. . .

handling

10.1.8 Handling natural transformations. By the Theorem, natural transformations between polynomial functors are diagrams like above. To compose them, one should just compose the corresponding natural transformations, and then factor the result into representable followed by cartesian. This is not an economical way to do it in practice though, because the starting representable and the ending cartesian factors are already in place. It is enough to refactor the middle part: refactor cartesian-followed-by-representable into representable-followed-by-cartesian. This is very easy: given

$$\begin{tikzcd}
 & \overline{E} \arrow[r, "\overline{p}"] \arrow[ld, "\overline{s}"'] \arrow[d, "v"] \arrow[rd, phantom, "\lrcorner", very near start] & \overline{B} \arrow[d, "u"'] \arrow[rd, "\overline{t}"] & \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] \arrow[d, equal] & J \\
 & F \arrow[lu, "z"] \arrow[u, "w"] \arrow[r, "q"'] & B \arrow[ru, "t"'] &
\end{tikzcd} \tag{10.1}$$

bad-order

define the composite to be

$$\begin{tikzcd}
 & \overline{E} \arrow[r, "\overline{p}"] \arrow[dl, "\overline{s}"'] & \overline{B} \arrow[d, equal] \arrow[dr, "\overline{t}"] & \\
I & \overline{F} \arrow[l, "\overline{z}"'] \arrow[u, "\overline{w}"'] \arrow[r, "\overline{q}"] \arrow[d, "f"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{B} \arrow[r, "\overline{t}"'] \arrow[d, "u"'] & J \\
 & F \arrow[ul, "z"] \arrow[r, "q"'] & B \arrow[ur, "t"'] &
\end{tikzcd} \tag{10.2}$$

good-order

where $\overline{F} := F \times_B \overline{B}$, and $\overline{w} : \overline{F} \to \overline{E}$ is induced by w and the universal property of $\overline{E}$. That this definition is correct is the content of the following lemma.

10.1.9 Lemma. *The two diagrams,* (10.1) *and* (10.2), *define the same natural transformation.*

Proof. The two diagrams fit together in a single diagram

$$\begin{tikzcd}
 & & \overline{E} \arrow[drr, "\overline{p}"] \arrow[dddll, "\overline{s}"'] \arrow[ddddr, "v"] & & & \\
 & & & & \overline{B} \arrow[dddd, "u"'] \arrow[ddr, "\overline{t}"] & \\
 & \overline{F} \arrow[uur, "\overline{w}"'] \arrow[dl, "\overline{z}"] \arrow[urrr, "\overline{q}"'] \arrow[ddddr, "f"] & & & & \\
I & & & & & J \\
 & & & E \arrow[ulll, "s"'] \arrow[dr, "p"] & & \\
 & & & & B \arrow[uur, "t"'] & \\
 & & F \arrow[uur, "w"'] \arrow[uuull, "z"] \arrow[urr, "q"'] & & &
\end{tikzcd}$$

in which the three squares are pullback squares. The statement of the lemma is the following equation of natural transformations (written in left-to-right composition):

$$\begin{tikzcd}
\overline{s}^*\overline{p}_*\overline{t}_! \arrow[r, Rightarrow, "\text{unit}"] \arrow[d, Rightarrow, "\text{BC}"'] & \overline{z}^*\overline{q}_*\overline{t}_! \arrow[d, Rightarrow, "\text{BC}"] \\
s^*p_*u^*u_!t_! \arrow[d, Rightarrow, "\text{counit}"'] & z^*q_*u^*u_!t_! \arrow[d, Rightarrow, "\text{counit}"] \\
s^*p_*t_! \arrow[r, Rightarrow, "\text{unit}"'] & z^*q_*t_!
\end{tikzcd}$$

The left-bottom composite is the natural transformation induced by (10.1), and the top-right composite is the one induced by (10.2). The two units are the representable transformations, while the composites BC-counit are the cartesian transformations. The proof consists in filling in with standard equations. Here is the full pasting diagram:

$$\begin{tikzcd}
\overline{s}^*\overline{p}_*\overline{t}_! =s^*v^*\overline{p}_*u_!t_! \arrow[rrr] \arrow[ddr] \arrow[ddd] & & & s^*v^*\overline{w}^*\overline{w}_*\overline{p}_*u_!t_! = \overline{z}^*\overline{q}_*\overline{t}_! \arrow[d, equal] \\
& & \text{A} & s^*w^*f^*\overline{w}_*\overline{p}_*u_!t_! \arrow[dll] \arrow[dd] \\
& s^*w^*w_*v^*\overline{p}_*u_!t_! \arrow[drr] & \text{B} & \\
s^*p_*u^*u_!t_! \arrow[rrr] \arrow[d] & & & s^*w^*w_*p_*u^*u_!t_! \arrow[d] \\
s^*p_*t_! \arrow[rrr] & & & s^*w^*w_*p_*t_! = z^*q_*t_!
\end{tikzcd}$$

Here region A is Lemma 8.4.1 (iv) applied to the square

$$\begin{tikzcd}
\overline{F} \arrow[r, "\overline{w}"] \arrow[d, "f"'] \arrow[dr, phantom, "\lrcorner", very near start] & \overline{E} \arrow[d, "v"] \\
F \arrow[r, "w"'] & E
\end{tikzcd}$$

and region B is Lemma 8.4.2 (iv) applied to the squares

$$\begin{tikzcd}
\overline{F} \arrow[r, "\overline{w}"] \arrow[d, "f"'] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{E} \arrow[r, "\overline{p}"] \arrow[d, "v"] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{B} \arrow[d, "u"] \\
F \arrow[r, "w"'] & E \arrow[r, "p"'] & B.
\end{tikzcd}$$

The two remaining squares are clearly commutative since each amounts to rewriting in two independent parts of the expressions. □

This gives an explicit description of composition of natural transformations between polynomial functors, but in order for it to be strictly associative (and strictly unital) we just need to recall that we identify two diagrams if they only differ in the middle part (this difference is unique, so there are no coherence problems). Indeed, two different factorisations of the same natural transformation obviously represent the same natural transformation. . .

10.1.10 Corollary. *With the vertical composition just introduced, the assignment* $\boldsymbol{Brdg}(I, J) \to \boldsymbol{Cat}(\boldsymbol{Set}/I, \boldsymbol{Set}/J)$ *is functorial.*

□

Together with **??** and **??** this establishes:

10.1.11 Lemma. *For any two fixed sets* $I, J \in \boldsymbol{Set}$*, the functor* $\boldsymbol{Brdg}(I, J) \to \boldsymbol{Cat}(\boldsymbol{Set}/I, \boldsymbol{Set}/J)$ *is fully faithful.* □

Here is another outcome of the theory, as far as I can see:

10.1.12 Proposition. *Every natural transformation between composites of upperstars, lowershrieks and lowerstars is a composite of units and counits of the two adjunctions, as well as the invertible* 2*-cells expressing pseudo-functoriality of pullback and its adjoints.*

Proof. A natural transformation is a composite of a representable transformation and a cartesian one. Both we have checked that they are given in terms of units and counits. □

Basic properties of $\boldsymbol{PolyFun}(I, J)$: sums and products

10.1.13 Sums. The sum of two polynomial functors (with common source and target) is just obtained by taking sums of B and E:

$$\begin{tikzcd} & E_1 + E_2 \arrow[r] \arrow[dl] & B_1 + B_2 \arrow[dr] & \\ I & & & J \end{tikzcd}$$

WRITE OUT THE DETAILS HERE. check that the universal property is satisfied. Check also that the sum inclusions are cartesian.

10.1.14 Products. When we talk about products of polynomial functors with values in $\boldsymbol{Set}_J$, we are referring of course to products in $\boldsymbol{Set}/J$. Note that if X and Y are sets over J then their product in $\boldsymbol{Set}/J$ is just the fibre product $X \times_J Y$.

The product of two polynomial functors should be just the product of their values. The product of $I \leftarrow E \rightarrow B \rightarrow J$ and $I \leftarrow F \rightarrow C \rightarrow J$ should be

$$\begin{tikzcd} E \times_J C + B \times_J F \arrow[rr] \arrow[d, "s"'] & & B \times_J C \arrow[d] \\ I & & J \end{tikzcd}$$

Here s is given by projection onto the E and F factors. (Note that this is not the product in the smaller category $\boldsymbol{Poly}^{\mathrm{c}}(I, J)$.)

SOME EXAMPLE

10.1.15 Extensivity. Just look at polynomial functors in one variable, but over an arbitrary lccc $\mathscr{E}$. IN ORDER FOR THE PRODUCT OF TWO POLYNOMIAL FUNCTORS TO BE POLYNOMIAL AGAIN, WE NEED SOME EXTRA ASSUMPTION. THE CLEANEST ASSUMPTION IS THAT $\mathscr{E}$ BE EXTENSIVE.

See handwritten notes.

Misc

Temporarily let $[n]$ denote the indexing set $\{1, 2, \ldots, n\}$. Let $\boldsymbol{FinSet}[X_1, \ldots, X_n]$ denote the category of polynomial functors $\boldsymbol{Set}/[n] \rightarrow \boldsymbol{Set}$ represented by $[n] \leftarrow E \to B$ with E and B finite. The Burnside semiring of this is $\mathbb{N}[X_1, \ldots, X_n]$.

$\boldsymbol{Poly}^{\mathrm{c}}(I,J)$: the cartesian fragment

Come back to the fibration in groupoids

$$\begin{array}{rcl} \boldsymbol{Poly}^{\mathrm{c}}(I,J) & \longrightarrow & \boldsymbol{Set}/J \\ P & \longmapsto & P(1) \end{array}$$

(note that it is a fibration in groupoids since by definition all the arrows upstairs are cartesian!) Take slice of it:

$$\begin{array}{rcl} \boldsymbol{Poly}^{\mathrm{c}}(I,J)/P & \longrightarrow & \boldsymbol{Set}/B \\ [Q \to P] & \longmapsto & [Q1 \to P1] \end{array}$$

This functor is an equivalence of categories.

... the key point is that there is a natural equivalence of categories

$$\boldsymbol{Poly}^{\mathrm{c}}(I)/P \xrightarrow{\sim} \boldsymbol{Set}/B, \tag{10.3}$$

club-eq

given by evaluation at the terminal object $I \to I$, which we denote by 1. In detail, if $Q \to P$ is an object in $\boldsymbol{Poly}^{\mathrm{c}}(I)/P$, the associated object in $\boldsymbol{Set}/B$ is simply $Q(1) \to P(1) = B$. The inverse equivalence basically takes an object $C \to B$ in $\boldsymbol{Set}/B$ to the object in $\boldsymbol{Poly}^{\mathrm{c}}(I)/P$ given by the fibre square

$$\begin{tikzcd}
 & E \times_B C \arrow[r] \arrow[d] & C \arrow[d] & \\
I & E \arrow[l] \arrow[r] & B \arrow[r] & I.
\end{tikzcd}$$

Sums and products in $\boldsymbol{Poly}^{\mathrm{c}}(I,J)$

NOT WRITTEN YET

10.2 Horizontal composition and the bicategory of polynomial functors

Define ***PolyFun*** to be the sub-2-category of ***Cat*** whose objects are the slices of ***Set***, whose arrows are the polynomial functors, and whose 2-cells are the natural transformations.

As defined, ***PolyFun*** is a strict 2-category. A key feature of the theory is that polynomial functors (and the natural transformations between them) can be represented by diagrams. The diagrams constitute a biequivalent non-strict bicategory which we now describe. The relation between the two is similar to theories versus models, or abstract manifolds versus coordinate charts. We wish to stress that the intrinsic objects are the actual functors: those functors that *admit* a polynomial representation. By *choosing* representations we can work with them easily, and get to grips with the combinatorics. In practice it is convenient to blur the distinction between the functors and the representing diagrams. In this section we do make a careful distinction, only to justify the blur. We do this by defining a bicategory of polynomial diagrams (bridges) and establishing a biequivalence with the above strict 2-category ***PolyFun***.

10.2.1 Proposition. *There is a bicategory* ***Brdg*** *whose objects are the objects of* ***Set****, whose arrows from I to J are the bridge diagrams like*

$$\begin{tikzcd}
 & B \arrow[r, "f"] \arrow[dl, "s"'] & A \arrow[dr, "t"] & \\
I & & & J
\end{tikzcd}$$

and whose 2*-cells are diagrams like*

$$\begin{tikzcd}
 & D \arrow[r, "g"] \arrow[dl, "u"'] & C \arrow[d, equal] \arrow[dr, "v"] & \\
I & D' \arrow[l] \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & C \arrow[r] \arrow[d] & J \\
 & B \arrow[r, "f"'] \arrow[ul] & A \arrow[ur] &
\end{tikzcd} . \tag{10.4}$$

Brdg-2cell

(Composition of bridges was described above, composition laws for 2*-cells are described in the proof below.) The assignment of polynomial functors and natural transformations to such diagrams as in* 10.1.4 *constitutes a biequivalence with the category* ***PolyFun****.*

The vertical composition was already treated. It remains to treat horizontal composition of 2-cells.

Note that the arrows are the polynomial functors $\mathbf{Set}/I \to \mathbf{Set}/J$, not the diagrams. This meant that we are including functors which are only isomorphic to ones given by such diagrams. (And also that we are considering two diagrams the same if they represent the same functor CAN THIS HAPPEN??? (and then they differ only by some bijections).) Doing it this way ensures that composition is well-defined and strictly associative. Had we taken the arrows to be diagrams, then the composition would only be defined up to unique isomorphisms (since it basically relies on pullback), and composition would only be defined up to this sort of sloppiness, and hence it would not be strictly associative. We would get only a bicategory. Even for 2-cells, if we had let the 2-cells be the actual diagrams we would have some looseness, suggesting that there might even be a weak 3-category around: the vertical composition of 2-cells is not strict, because it involves a pullback too. (Note however that in this case, one could get away with defining two 2-cells to be the same if there is a comparison isomorphism, and since this is a comparison between two pullbacks there can be at most one, and hence there is a clique of equivalent 2-cells. . .)

The second ingredient we need is the following observation that we already established:

thm:cohbicat **10.2.2 Lemma.** *The assignment of polynomial functors to bridges is compatible with composition (up to specified isomorphisms), and sends identity bridges to identity functors (up to specified isomorphisms). More precisely,*

(i) For every pair of bridges $F : I \to I'$ and $F' : I' \to I''$, we have a natural isomorphism

$$\phi_{F',F} : \widetilde{F'} \circ \widetilde{F} \Rightarrow \widetilde{F'F} \,.$$

(ii) For every object $I \in \mathbf{Set}$, we have a natural isomorphism

$$\phi_I : \mathsf{Id}_{\mathbf{Set}/I} \Rightarrow \widetilde{\mathrm{id}\, I} \,.$$

Some preliminary exercises in the cartesian fragment

Let $\boldsymbol{PolyFun}^{\mathrm{c}}$ denote the 2-category of all polynomial functors and their cartesian natural transformations. It is defined as a sub-2-category of ***Cat*** as follows: The objects are the slice categories $\textbf{\textit{Set}}/I$. The arrows from $\textbf{\textit{Set}}/I$ to $\textbf{\textit{Set}}/J$ are the polynomial functors (represented by diagrams $I \leftarrow E \rightarrow B \rightarrow J$), and the 2-cells are the natural transformations between them.

We should now explain how they are composed horizontally.

10.2.3 Some exercises. How to whisker a cartesian natural transformation with a polynomial functor.

The following long computation is like a first step in this direction, showing how to whisker a 2-cell (just a cartesian one) with a 1-cell. The horizontal composition could be defined in terms of this, but then we would have to verify that the two ways of defining them coincide. Also, we would have to generalise the argument to more general 2-cells than just the cartesian ones. . .

We will show that composition with a polynomial functor is functorial with respect to cartesian 2-cells. THIS SHOULD FOLLOW AUTOMATICALLY FROM GENERAL PRINCIPLES. . . YES, SINCE POLYNOMIAL FUNCTORS PRESERVE PULLBACKS (1.7.1), HORIZONTAL COMPOSITION OF CARTESIAN NATURAL TRANSFORMATIONS IS AGAIN CARTESIAN (2.5.2). Hence this long computation is probably superfluous. However, its ideas are needed for the double-cat stuff. . .

Given a polynomial functor

$$\begin{tikzcd}
& E \arrow[ld, "s"'] \arrow[rr, "p"] & & B \arrow[rd, "t"] & \\
I & & P & & J
\end{tikzcd}$$

then there is induced a map

$$\begin{array}{ccc}
\textbf{\textit{Poly}}(H,I) & \longrightarrow & \textbf{\textit{Poly}}(H,J) \\
Q & \longmapsto & P \circ Q
\end{array}$$

What we'll show here is that if there is given a cartesian natural transformation $Q \Rightarrow Q'$ (given by

$$\begin{tikzcd}
& F \arrow[ld] \arrow[r, "q"] \arrow[d] \arrow[rd, phantom, "\lrcorner", very near start] & C \arrow[rd] \arrow[d] & \\
H & F' \arrow[l] \arrow[r] & C' \arrow[r] & I
\end{tikzcd}$$

then there is induced a cartesian transformation $P \circ Q \Rightarrow P \circ Q'$.

The proof consists in splitting the statement into three pieces, depending on the three steps of P. It is enough to show that the result is true for these three functors

$$\textbf{\textit{Poly}}(H,I) \xrightarrow{s^*} \textbf{\textit{Poly}}(H,E) \xrightarrow{p_*} \textbf{\textit{Poly}}(H,B) \xrightarrow{t_!} \textbf{\textit{Poly}}(H,J)$$

For the first case, we are concerned with a diagram

$$\begin{tikzcd}
& F \arrow[ld] \arrow[r, "q"] \arrow[d] \arrow[rd, phantom, "\lrcorner", very near start] & C \arrow[rd] \arrow[d] & & \\
H & F' \arrow[l] \arrow[r] & C' \arrow[r] & I & E \arrow[l, "s"']
\end{tikzcd}$$

To compute the composition of these polynomial functors amounts to performing four pullbacks. In fact the resulting polynomial functor is just the pullback of the whole diagram along s:

$$\begin{tikzcd}
& F \times_I E \arrow[ld] \arrow[r] \arrow[d] \arrow[rd, phantom, "\lrcorner", very near start] & C \times_I E \arrow[rd] \arrow[d] & \\
H & F' \times_I E \arrow[l] \arrow[r] & C' \times_I E \arrow[r] & E
\end{tikzcd}$$

It is obvious that this square is again cartesian.

The second part is the hard part. Renaming the objects, we are in this situation:

$$\begin{tikzcd}
& F \arrow[r, "q"] \arrow[dl] \arrow[dd] \arrow[ddr, phantom, "\lrcorner", very near start] & C \arrow[dr] \arrow[dd] & & \\
H & & & E \arrow[r, "p"] & B \\
& F' \arrow[ul] \arrow[r] & C' \arrow[ur] & &
\end{tikzcd}$$

To compute this composition (we look at the top part), we first have to take $p_*(C)$. Now consider the pullback back along p:

$$\begin{tikzcd}
p^*p_*(C) \arrow[r] \arrow[d] & p_*(C) \arrow[d] \\
E \arrow[r, "p"'] & B
\end{tikzcd}$$

Now by adjunction we have the canonical evaluation map $p^*p_*(C) \to C$ (and we should check that it commutes with the map down to E). Description of this map: Now compare with what happens in the bottom part: the same thing. We have a map $p_*(C) \to p_*(C')$, and pulling it back we also get $p^*p_*(C) \to p^*p_*(C')$. This square

$$\begin{tikzcd}
p^*p_*(C) \arrow[r] \arrow[d] & p_*(C) \arrow[d] \\
p^*p_*(C') \arrow[r] & p_*(C')
\end{tikzcd}$$

is a pullback for the following reason: arrange the squares like this:

$$\begin{tikzcd}
p^*p_*C \arrow[r] \arrow[d] & p_*C \arrow[d] \\
p^*p_*C' \arrow[r] \arrow[d] & p_*C' \arrow[d] \\
E \arrow[r, "p"'] & B.
\end{tikzcd}$$

Here the big square and the bottom square are pullbacks by construction. Hence, the top square is too (A.1).

Next we pullback $p^*p_*(C) \to C$ along q and q'. This gives $q^*p^*p_*(C)$ and $q'^*p^*p_*(C')$. There is a map between them. Now we have cartesian squares

$$\begin{tikzcd}
q^*p^*p_*(C) \arrow[r] \arrow[d] & p^*p_*(C) \arrow[d] \\
F \arrow[r] \arrow[d] & C \arrow[d] \\
F' \arrow[r] & C' \\
q'^*p^*p_*(C') \arrow[r] \arrow[u] & p^*p_*(C') \arrow[u]
\end{tikzcd}$$

By some cube argument, we can conclude that the square

$$\begin{tikzcd}
q^*p^*p_*(C) \arrow[r] \arrow[d] & p^*p_*(C) \arrow[d] \\
q'^*p^*p_*(C') \arrow[r] & p^*p_*(C')
\end{tikzcd}$$

is also cartesian. This is what we wanted to prove.

(The cube argument is something like: if the left face bottom and front are cartesian then the top is too. This is just a variation of the previous general result about pullback squares.)

Finally for $t_!$ it is immediate that the cartesian squares are preserved, because it stays the same.

Note also that all of these three functors preserve monomorphisms. That is, if the vertical maps in the original pullback square are monos, then the resulting ones are too.

This is just a question of following the constructions through, and observe that we are mostly using pullback, and forming pullback preserves monos. At one point we also used a pushforth, but this is like taking product, and a product of monos is again a mono.

SLOGAN:

The slogan is that ALL 2-cells between polynomial functors are built from the units and counits of the adjunctions.

Horizontal composition of 2-cells

subsec:horizontal

A diagram

$$\begin{tikzcd}
 & \cdot \arrow[r] \arrow[dl] & \cdot \arrow[d, equal] \arrow[dr] & & \cdot \arrow[r] \arrow[dl] & \cdot \arrow[d, equal] \arrow[dr] & \\
\cdot \arrow[dr] & \cdot \arrow[l] \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] & \cdot \arrow[dr] & \cdot \arrow[l] \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] & \cdot \\
 & \cdot \arrow[r] & \cdot \arrow[ur] & & \cdot \arrow[r] & \cdot \arrow[ur] &
\end{tikzcd}$$

represents two horizontally composable natural transformations. The composite is again a natural transformation between polynomial functors, hence we know it can be represented by a diagram. We shall now give a direct construction of the diagram representing the horizontal composite. We shall take advantage of the interchange law to break the construction into two steps: first we compose the upper parts (the representable transformations) horizontally, and then we compose the lower parts (the cartesian transformations) horizontally. The lower part is rather straightforward: the horizontal composite of two cartesian transformations is again cartesian, and it is not difficult to construct the diagram. The representable part is somewhat more complicated: it turns out the horizontal composite of two representable transformations is not again representable in general. We analyse this now.

Given a general horizontal composite of representable transformations

$$\begin{tikzcd}
 & \cdot \arrow[dl] \arrow[dr] & & & \cdot \arrow[dl] \arrow[dr] & & \\
\cdot & & \cdot \arrow[r] & \cdot & & \cdot \arrow[r] & \cdot \\
 & \cdot \arrow[ul] \arrow[uu] \arrow[ur] & & & \cdot \arrow[ul] \arrow[uu] \arrow[ur] & &
\end{tikzcd}$$

we know how to construct the composites of each of the functors (8.7), and we shall relate those two big diagrams by connecting maps to establish a natural transformation between them which we then show represents the horizontal composite in the functor category. Taking advantage once again of the interchange law, we break the problem into two steps like this:

$$\begin{tikzcd}
& \cdot \arrow[dl] \arrow[dr] & & & \cdot \arrow[dl] \arrow[dr] & & \\
\cdot & & \cdot \arrow[r] & \cdot & & \cdot \arrow[r] & \cdot \\
& \cdot \arrow[ul] \arrow[ur] \arrow[uu] & & & & &
\end{tikzcd}$$

$$\begin{tikzcd}
& & & & \cdot \arrow[dl] \arrow[dr] & & \\
\cdot & & \cdot \arrow[r] & \cdot & & \cdot \arrow[r] & \cdot \\
& \cdot \arrow[ul] \arrow[ur] & & & \cdot \arrow[ul] \arrow[ur] \arrow[uu] & &
\end{tikzcd}$$

(and not the other way around).

The upper part, a representable natural transformation post-whiskered by a polynomial functor, is quite easy, and is again a representable natural transformation. We omit that construction.

The lower part, a representable natural transformation pre-whiskered with a polynomial functor, turns out to be complicated and is not again a representable natural transformation.

The interaction between upperstars and lowerstars is quite well-behaved, and so is the interaction between lowershriek and upperstar—as one can expect from Beck-Chevalley. The difficult case to handle is the interaction between lowershriek and lowerstar, which involves distributivity. In fact it is enough to describe how to normalise the 2-cell

$$\begin{tikzcd}
& & \cdot \arrow[dl, "z"'] \arrow[dr, "q"] & \\
\cdot \arrow[r, "c"] & \cdot & & \cdot \\
& & \cdot \arrow[ul, "\tilde{z}"] \arrow[ur, "\tilde{q}"'] \arrow[uu, "f"'] &
\end{tikzcd}$$

i.e. how to pre-whisker a representable natural transformation with a lowershriek functor. Recall that the natural transformation is (written left-to-right)

$$c_! z^* q_* \overset{\text{unit}}{\Rightarrow} c_! z^* f^* f_* q_* = c_! \tilde{z}^* f_* q_*$$

We will now work towards bringing it on normal form. The first step involves the interaction between $c_!$ and the triangle $\tilde{z} = fz$, so for the moment we can forget about q. So pull back the triangle $\tilde{z} = fz$ along c to get

$$\begin{tikzcd}
 & \cdot \arrow[rr, "s"] \arrow[dl, "k"'] & & \cdot \arrow[dl, "z"'] \\
\cdot & & \cdot \arrow[ll, "c"'] & \\
 & \cdot \arrow[uu, "n"'] \arrow[ul, "\tilde{k}"] \arrow[rr, "\tilde{s}"'] & & \cdot \arrow[uu, "f"'] \arrow[ul, "\tilde{z}"]
\end{tikzcd}$$

as prescribed in the normalisation procedure (Proof of Lemma 8.5.1). We now have the following diagram of natural transformations, comparing the original natural transformation with one step of normalisation via Beck-Chevalley isomorphisms.

$$\begin{tikzcd}
c_! z^* \arrow[d, Rightarrow, "\text{unit}"'] & \overset{\text{BC}}{\simeq} & k^* s_! \arrow[d, Rightarrow, "\text{unit}"] \\
c_! z^* f^* f_* \arrow[d, equal] & \overset{\text{BC}}{\simeq} & k^* s_! f^* f_* \arrow[d, Rightarrow, "\text{BC}"] \\
c_! \tilde{z}^* f_* & \overset{\text{BC}}{\simeq} & k^* n^* \tilde{s}_! f_*
\end{tikzcd}$$

Here composition is written left-to-right. The top square is obviously commutative. The bottom 'square' is commutative by Lemma 8.4.2 (the fact that Beck-Chevalley along a composite can be performed in two steps).

For the next step in the normalisation rewriting, there is a trailing q_*, but in contrast we no longer have to care about the leading k^*, and we can also forget about the maps c, z, and $\tilde{z}$. So the situation is now the diagram

$$\begin{tikzcd}
\widetilde{N} \arrow[r, "n"] \arrow[d, "\tilde{s}"'] \arrow[dr, phantom, "\lrcorner", very near start] & N \arrow[d, "s"] & \\
\widetilde{F} \arrow[r, "f"'] & F \arrow[r, "q"'] & C
\end{tikzcd}$$

and a natural transformation (composition written left-to-right)

$$s_! q_* \overset{\text{unit}}{\Rightarrow} s_! f^* f_* q_* \overset{\text{BC}}{\simeq} n^* \tilde{s}_! (fq)_* .$$

Both these functors are now rewritten using distributivity to acquire normal form. The relevant diagram for this rewriting is the following:

$$\begin{tikzcd}
W \arrow[r, "r"] \arrow[d, "u"'] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[r, "p"] \arrow[dd, "e"] \arrow[dr, phantom, "\lrcorner", very near start] & D \arrow[d, "v"] \\
\widetilde{E} \arrow[rr, "\tilde{p}", crossing over] \arrow[d, "\tilde{e}"'] \arrow[dr, phantom, "\lrcorner", very near start] & & \widetilde{D} \arrow[dd, "t"] \\
\widetilde{N} \arrow[r, "n"] \arrow[d, "\tilde{s}"'] \arrow[dr, phantom, "\lrcorner", very near start] & N \arrow[d, "s"] & \\
\widetilde{F} \arrow[r, "f"'] & F \arrow[r, "q"'] & C
\end{tikzcd}$$

Here $D = q_* N$, and $E = q^* q_* N$ (i.e. the right-hand square is a distributivity square). Similarly, $\widetilde{D} = (q \circ f)_* \widetilde{N}$, and $\widetilde{E} = (q \circ f)^* (q \circ f)_* \widetilde{N}$. The map v is induced as follows (written right-to-left, since if involves evaluation on objects):

$$\widetilde{D} = (q \circ f)_* \widetilde{N} = q_* f_* f^* N \xleftarrow{\text{unit}} q_* N = D.$$

The diagram includes the diagram for a natural transformation between polynomial functors on normal form:

$$\begin{tikzcd}
 & E \arrow[r, "p"] \arrow[dl, "e"'] & D \arrow[d, equal] \arrow[dr] & \\
N & W \arrow[l] \arrow[u, "r"] \arrow[r] \arrow[d, "u"'] \arrow[dr, phantom, "\lrcorner", very near start] & D \arrow[r] \arrow[d, "v"] & C \\
 & \widetilde{E} \arrow[r, "\tilde{p}"'] \arrow[ul, "\tilde{e}n"] & \widetilde{D} \arrow[ur, "t"'] &
\end{tikzcd}$$

As usual, the induced natural transformation is (composition written left-to-right)

$$\begin{aligned} e^*p_*v_!t_! &\Rightarrow e^*r^*r_*p_*v_!t_! \\ &= n^*\tilde{e}^*u^*r_*p_*v_!t_! \\ &\simeq n^*\tilde{e}^*\tilde{p}_*v^*v_!t_! \\ &\Rightarrow n^*\tilde{e}^*\tilde{p}_*t_! \end{aligned}$$

consisting of a unit (the representable part), followed by a Beck-Chevalley and a counit (the cartesian part).

The claim is that this natural transformation is precisely the result of conjugating the original natural transformation by distributivity. In other words, the following diagram of 2-cells commutes (composition written left-to-right):

$$\begin{tikzcd}
s_!q_* \arrow[rr, Leftrightarrow, "\text{distr.}"] \arrow[dd, Rightarrow, "\text{unit}"'] & & \boxed{e^*p_*v_!t_!} \arrow[d, Rightarrow, "\text{unit}"] \\
& & e^*r^*r_*p_*v_!t_! \arrow[d, equal] \\
s_!f^*f_*q_* \arrow[dd, Rightarrow, "\text{BC}"'] & & \boxed{n^*\tilde{e}^*u^*r_*p_*v_!t_!} \arrow[d, Rightarrow, "\text{BC}"] \\
& & n^*\tilde{e}^*\tilde{p}_*v^*v_!t_! \arrow[d, Rightarrow, "\text{counit}"] \\
n^*\tilde{s}_!(fq)_* \arrow[rr, Rightarrow, "\text{distr.}"] & & \boxed{n^*\tilde{e}^*\tilde{p}_*t_!}
\end{tikzcd}$$

The left-hand column is the natural transformation we started with. The right-hand column is the natural transformation obtained from the diagram that serves to apply distributivity to the original functors to get them on normal form—the boxed functors are those on normal form and constitute the factorisation into representable followed by cartesian. The statement is that the resulting natural transformation is precisely the one obtained from distributivity.

The proof consists in filling the interior of the diagram with standard 2-cells. Note that the distributivity isomorphism factors as a counit followed by lowershriek-lowerstar exchange. Neither of these are invertible, so the verification must keep track of directions of arrows. Here is the relevant pasting diagram (with composition written left-to-right):

$$\begin{tikzcd}
 & s_! q_* \arrow[dddl] & & e^* e_! s_! q_* \arrow[ll] \arrow[ddl] \arrow[ddr] & & e^* p_* v_! t_! \arrow[ll] \arrow[ddr] & & & \\
 & & & \text{A} & & & & & \\
 & & e^* e_! s_! f^* f_* q_* \arrow[dll] \arrow[ddddl] \arrow[dddr] & & e^* r^* r_* e_! s_! q_* \arrow[dddl] & & e^* r^* r_* p_* v_! t_! \arrow[ll] \arrow[dddlll] \arrow[dddl] \arrow[ddr] & & \\
s_! f^* f_* q_* \arrow[ddddd] & & & & \text{E} & & & & \\
 & & \text{D} & & & \text{F} & & n^* \tilde{e}^* u^* r_* p_* v_! t_! \arrow[dddl] \arrow[ddr] & \\
 & & & e^* r^* u_! \tilde{e}_! \tilde{s}_! f_* q_* \arrow[dd] & & e^* r^* u_! \tilde{p}_* t_! \arrow[ll] \arrow[ddr] & & & \\
 & e^* e_! n^* \tilde{s}_! f_* q_* \arrow[urr] \arrow[ddl] & & & & & & & n^* \tilde{e}^* \tilde{p}_* v^* v_! t_! \arrow[dddl] \\
 & \text{B} & & n^* \tilde{e}^* u^* u_! \tilde{e}_! \tilde{s}_! f_* q_* \arrow[dlll] \arrow[ddl] & & & n^* \tilde{e}^* u^* u_! \tilde{p}_* t_! \arrow[lll] \arrow[ddr] & \text{C} & \\
n^* \tilde{s}_! f_* q_* & & & & & & & & \\
 & & n^* \tilde{e}^* \tilde{e}_! \tilde{s}_! f_* q_* \arrow[ull] & & & & & n^* \tilde{e}^* \tilde{p}_* t_! \arrow[lllll] &
\end{tikzcd}$$

Here region A is Lemma 8.4.1 (xi) applied to the square

$$\begin{tikzcd}
\cdot \arrow[r, "r"] \arrow[d, "u\tilde{e}\tilde{s}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "es"] \\
\cdot \arrow[r, "f"'] & \cdot
\end{tikzcd}$$

Region B is Lemma 8.4.1 (iii) applied to the square

$$\begin{tikzcd}
\cdot \arrow[r, "r"] \arrow[d, "u\tilde{e}"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "e"] \\
\cdot \arrow[r, "n"'] & \cdot
\end{tikzcd}$$

Region C is Lemma 8.4.1 (xii) applied to the square

$$\begin{tikzcd}
\cdot \arrow[r, "rp"] \arrow[d, "u"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "v"] \\
\cdot \arrow[r, "\tilde p"'] & \cdot
\end{tikzcd}$$

Region D is Lemma 8.4.2 (i) applied to the squares

$$\begin{tikzcd}
\cdot \arrow[r, "r"] \arrow[d, "u\tilde e"'] & \cdot \arrow[d, "e"] \\
\cdot \arrow[r, "n"'] \arrow[d, "\tilde s"'] & \cdot \arrow[d, "s"] \\
\cdot \arrow[r, "f"'] & \cdot
\end{tikzcd}$$

Region E is Lemma 8.4.2 (v) applied to the squares

$$\begin{tikzcd}
\cdot \arrow[r, "r"] \arrow[d, "u\tilde e\tilde s"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r, "p"] \arrow[d, "es"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "vt"] \\
\cdot \arrow[r, "f"'] & \cdot \arrow[r, "q"'] & \cdot
\end{tikzcd}$$

Region F is Lemma 8.4.2 (vi) applied to the squares

$$\begin{tikzcd}
\cdot \arrow[r, "rp"] \arrow[d, "u"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "v"] \\
\cdot \arrow[r, "\tilde p"'] \arrow[d, "\tilde e\tilde s"'] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "t"] \\
\cdot \arrow[r, "fq"'] & \cdot
\end{tikzcd}$$

The other regions are commutative for obvious reasons: the squares are commutative because they amount to rewriting in two independent parts of the expression. The triangle in the lower left-hand corner is commutative because the counit of a composite adjunction is the composite of the counits.

Chapter 11

Double categories of polynomial functors

The material of this chapter is mostly from Gambino-Kock [39].

11.1 Summary

It is important to be able to compare polynomial functors with different endpoints, and to base change polynomial functors along maps in ***Set***. This need can been seen already for linear functors: a small category can be seen as a monad in the bicategory of spans, but in order to get functors between categories with different object sets, one needs maps between spans with different endpoints. There are two ways to account for this: either in terms of a bicategory with extra structure [105], or in terms of a double category with certain properties: existence of companions and conjoints [45], folding [36], or the fibration property identified by Shulman [97], who uses the term *framed bicategories* for double categories having the property. We shall take the double-category viewpoint, mostly following Shulman, as this allows for diagrammatic representation.

The double categories of polynomial functors generalise module-like double categories, like the double category of spans.

There will be two double categories: one with general 2-cells, and one with only cartesian 2-cells.

11.1.1 The general double category. On the level of diagrams (which we call the intensional level), the double category of polynomials (denoted $\boldsymbol{Poly}_\square$) will have 2-cells of the form

$$\begin{tikzcd}
I' \arrow[dd, "u"'] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] \arrow[d, equal] & J' \arrow[dd, "v"] \\
& \cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] & \\
I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd} \qquad (11.1)$$

general2cell

i.e. like the diagrams defining natural transformations, but with the endpoints opened up. The extensional version of this double category (denoted $\boldsymbol{PolyFun}_\square$, involving the polynomial functors) will have 2-cells of the form

$$\begin{tikzcd}
\boldsymbol{Set}/I' \arrow[r, "P'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\swarrow"] & \boldsymbol{Set}/J' \arrow[d, "v_!"] \\
\boldsymbol{Set}/I \arrow[r, "P"'] & \boldsymbol{Set}/J
\end{tikzcd} \qquad (11.2)$$

It follows from standard arguments (cf. the proof below) that $\boldsymbol{PolyFun}_\square$ is a framed bicategory. Our main task is to translate this to the level of diagrams, and make explicit how the various operations are performed at this level.

11.1.2 Cartesian fragment. The double category with only cartesian 2-cells, which we denote $\boldsymbol{Poly}^{\mathrm{c}}_\square$ has sets as objects and set maps as vertical arrows. It has polynomial functors as horizontal arrows, and its squares are diagrams

$$\begin{tikzcd}
D \arrow[d] & U \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & V \arrow[r] \arrow[d] & C \arrow[d] \\
I & E \arrow[l] \arrow[r] & B \arrow[r] & J.
\end{tikzcd}$$

The extension of such a diagram is precisely a cartesian natural transformation

$$\begin{tikzcd}
\textbf{\textit{Set}}/I' \arrow[r, "P'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\swarrow"] & \textbf{\textit{Set}}/J' \arrow[d, "v_!"] \\
\textbf{\textit{Set}}/I \arrow[r, "P"'] & \textbf{\textit{Set}}/J
\end{tikzcd} \tag{11.3}$$

$\boxed{\texttt{sw-cart}}$

which are the 2-cells of a equivalent double category denoted $\textbf{\textit{PolyFun}}^{\mathrm{c}}_{\square}$

The cartesian fragment is the most important for applications. In particular we shall be concerned with applications to operads. When the polynomial functors are polynomial monads corresponding to coloured operads, the notion (of monad maps of the above kind) agrees with the notion of morphism between coloured operads via base change.

Reminder on double categories

WRITE THIS SECTION

11.1.3 Double categories. Recall that a double category is a category internal to ***Cat***. Spelled out this means a double category $\mathbb{D}$ has a category of objects $\mathbb{D}_0$, a category of morphisms $\mathbb{D}_1$, together with structure functors

$$\begin{tikzcd}
\mathbb{D}_0 \arrow[r, shift left=2, leftarrow, "\partial_1"] \arrow[r] \arrow[r, shift right=2, leftarrow, "\partial_0"'] & \mathbb{D}_1 & \mathbb{D}_1 \times_{\mathbb{D}_0} \mathbb{D}_1 \arrow[l, "\text{comp.}"']
\end{tikzcd}$$

subject to usual category axioms. The objects of $\mathbb{D}_0$ are called *objects* of $\mathbb{D}$, the morphisms of $\mathbb{D}_0$ are called *vertical arrows*, the objects of $\mathbb{D}_1$ are called *horizontal arrows*, and the morphisms of $\mathbb{D}_1$ are called *squares*. For details, see [97], [45], [22]. As is custom, we allow for the possibility that the horizontal composition is associative and unital only up to specified coherent isomorphisms. Precisely, a double category is a pseudo-category [80] in the 2-category CAT; see also [75, §5.2].

In practice one often constructs $\mathbb{D}_1$ first and then specify the composition law to produce the double category. We use the convention that if **SomeCat** denotes a category playing the role of $\mathbb{D}_1$, we put a small square decoration to refer to the whole double category, $\mathbf{SomeCat}_{\square}$.

11.1.4 Framed bicategories. A *framed bicategory* [97] is a double category for which $(\partial_0, \partial_1) : \mathbb{D}_1 \longrightarrow \mathbb{D}_0 \times \mathbb{D}_0$ is a bifibration. (In fact, if it is a fibration then it is automatically an opfibration, and vice versa.) The upshot of this condition is that horizontal arrows (i.e. objects of $\mathbb{D}_1$) can be base changed back along arrows in $\mathbb{D}_0 \times \mathbb{D}_0$ (i.e. pairs of vertical arrows), and cobase changed forth along arrows in $\mathbb{D}_0 \times \mathbb{D}_0$.

11.1.5 Terminology. We need to fix some terminology. The characteristic property of a fibration is that every arrow in the base category admits a cartesian lift, and that every arrow in the total space factors (essentially uniquely) as a vertical arrow followed by a cartesian one. In the present situation, the term 'cartesian' is already in use to designate cartesian natural transformations (which fibrationally speaking are vertical rather than cartesian!), and the term 'vertical' already has a meaning in the double-category setting. For these reasons, instead of saying 'cartesian arrow' for a fibration we shall say *transporter* arrow; this terminology goes back to Grothendieck [46]. Correspondingly we shall say *cotransporter* instead of opcartesian. We shall simply refrain from using 'vertical' in the fibration sense: the arrows mapping to identity arrows by the fibration will be precisely the natural transformations of polynomial functors.

The double category of polynomial functors

11.1.6 The double category of polynomial functors. We want to extend the bicategories ***Poly*** and ***PolyFun*** to double categories. The objects of the double category ***PolyFun***$_\square$ are the slices of ***Set***, and the horizontal arrows are the polynomial functors. The vertical arrows are the lowershriek functors, and the squares in ***PolyFun***$_\square$ are of the form

$$\begin{tikzcd}
\boldsymbol{Set}/I' \arrow[r, "P'"] \arrow[d, "u_!"'] & \boldsymbol{Set}/J' \arrow[d, "v_!"] \arrow[dl, Rightarrow, shorten=12mm, "\phi"] \\
\boldsymbol{Set}/I \arrow[r, "P"'] & \boldsymbol{Set}/J
\end{tikzcd} \tag{11.4}$$

SW

where P' and P are polynomial functors and ϕ is a natural transformation.

Before describing the diagrammatic version of this double category, it is very helpful to have its framed structure available, so we establish that first.

thm:frabic **11.1.7 Proposition.** *The double category* $\boldsymbol{PolyFun}_{\square}$ *is a framed bicategory.*

Proof. The claim is that the 'endpoints' functor

$$\begin{array}{rcl} \boldsymbol{PolyFun}_{\square} & \longrightarrow & \boldsymbol{Set} \times \boldsymbol{Set} \\ [P : \boldsymbol{Set}/I \to \boldsymbol{Set}/J] & \longmapsto & (I, J) \end{array}$$

is a bifibration. For each pair of arrows $(u, v) \in \boldsymbol{Set} \times \boldsymbol{Set}$ we have the following basic squares (companion pairs and conjoint pairs):

$$\begin{tikzcd} \cdot \arrow[r, "u_!"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow\mathrm{id}"] & \cdot \arrow[d, equal] \\ \cdot \arrow[r, equal] & \cdot \end{tikzcd} \qquad \begin{tikzcd} \cdot \arrow[r, equal] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow\eta"] & \cdot \arrow[d, equal] \\ \cdot \arrow[r, "u^*"'] & \cdot \end{tikzcd} \qquad \begin{tikzcd} \cdot \arrow[r, equal] \arrow[d, equal] \arrow[dr, phantom, "\Swarrow\mathrm{id}"] & \cdot \arrow[d, "v_!"] \\ \cdot \arrow[r, "v_!"'] & \cdot \end{tikzcd} \qquad \begin{tikzcd} \cdot \arrow[r, "v^*"] \arrow[d, equal] \arrow[dr, phantom, "\Swarrow\varepsilon"] & \cdot \arrow[d, "v_!"] \\ \cdot \arrow[r, equal] & \cdot \end{tikzcd} \tag{11.5}$$

It is now direct to check that the pasted square

$$\begin{tikzcd} \boldsymbol{Set}/I' \arrow[r, "u_!"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow\mathrm{id}"] & \boldsymbol{Set}/I \arrow[r, "P"] \arrow[d, equal] \arrow[dr, phantom, "\Swarrow\mathrm{id}"] & \boldsymbol{Set}/J \arrow[r, "v^*"] \arrow[d, equal] \arrow[dr, phantom, "\Swarrow\varepsilon"] & \boldsymbol{Set}/J' \arrow[d, "v_!"] \\ \boldsymbol{Set}/I \arrow[r, equal] & \boldsymbol{Set}/I \arrow[r, "P"'] & \boldsymbol{Set}/J \arrow[r, equal] & \boldsymbol{Set}/J \end{tikzcd}$$

is a transporter lift (cartesian lift) of (u, v) to P. We call $v^* \circ P \circ u_!$ the *base change* of P along (u, v), and denote it $(u, v)^\sharp(P)$. Dually, it is direct to check that the pasted square

$$\begin{tikzcd} \boldsymbol{Set}/I' \arrow[r, equal] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow\eta"] & \boldsymbol{Set}/I' \arrow[r, "P'"] \arrow[d, equal] \arrow[dr, phantom, "\Swarrow\mathrm{id}"] & \boldsymbol{Set}/J' \arrow[r, equal] \arrow[d, equal] \arrow[dr, phantom, "\Swarrow\mathrm{id}"] & \boldsymbol{Set}/J' \arrow[d, "v_!"] \\ \boldsymbol{Set}/I \arrow[r, "u^*"'] & \boldsymbol{Set}/I' \arrow[r, "P'"'] & \boldsymbol{Set}/J' \arrow[r, "v_!"'] & \boldsymbol{Set}/J \end{tikzcd}$$

is a cotransporter lift (opcartesian lift) of (u, v) to P'. We call $v_! \circ P' \circ u^*$ the *cobase change* of P' along (u, v), and denote it $(u, v)_\sharp(P')$. □

11.1.8 Remark. The above procedure of getting a framed bicategory out of a bicategory is a general construction: starting from a bicategory $\mathscr{C}$ and a subcategory $\mathscr{L}$ (comprising all the objects), there is a double category whose horizontal arrows are all the morphisms, whose vertical arrows are those from $\mathscr{L}$, and whose 2-cells are those of $\mathscr{C}$. If furthermore every morphism in $\mathscr{L}$ is a left adjoint, then the constructed double category is a framed bicategory. For details, see [97, Appendix].

Via the biequivalence $\boldsymbol{Poly} \simeq \boldsymbol{PolyFun}$ between the bicategory of polynomials and the 2-category of polynomial functors, Proposition 11.1.7 gives us also a framed bicategory of polynomials $\boldsymbol{Poly}_\square$, featuring nice diagrammatic representations which we now spell out, extending the results of Chapter 10. The following proposition is the double-category version of Proposition 10.1.4.

11.1.9 Proposition. *The squares (11.4) of $\boldsymbol{PolyFun}_\square$ are represented by diagrams of the form*

$$\begin{tikzcd}
P' : & I' \arrow[dd, "u"'] & B' \arrow[l] \arrow[r] & A' \arrow[r] \arrow[d, equal] & J' \arrow[dd, "v"] \\
 & & \cdot \arrow[u] \arrow[d] \arrow[r] \arrow[dr, phantom, very near start, "\lrcorner"] & \cdot \arrow[d] & \\
P : & I & B \arrow[l] \arrow[r] & A \arrow[r] & J .
\end{tikzcd} \tag{11.6}$$

equ:sqdiag

This representation is unique up to choice of pullback in the middle.

Proof. By Proposition 10.1.4, diagrams like (11.6) (up to choice of pullback) are in bijective correspondence with natural transformations $v_! \circ P' \circ u^* \Rightarrow P$, which by adjointness correspond to strong natural transformations $v_! \circ P' \Rightarrow P \circ u_!$, i.e. squares (11.4) in $\boldsymbol{PolyFun}_\square$. □

We can now conclude the following corollary, although we'll have to unravel the structure of framed bicategory implied on $\boldsymbol{Poly}_\square$:

11.1.10 Corollary. *Extension constitutes a framed biequivalence*

$$\boldsymbol{Poly}_\square \xrightarrow{\sim} \boldsymbol{PolyFun}_\square .$$

11.1.11 The double category of polynomials (bridge diagrams). We define a category $BRDG$ whose objects are bridges

$$\begin{tikzcd} I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J \end{tikzcd},$$

and whose arrows are diagrams

$$\begin{tikzcd}
I' \arrow[dd, "u"'] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] \arrow[d, equal] & J' \arrow[dd, "v"] \\
 & \cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] & \\
I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd}$$

The vertical composition of two diagrams

$$\begin{tikzcd}
\cdot \arrow[dd] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] \arrow[d, equal] & \cdot \arrow[dd] \\
 & \cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] & \\
\cdot \arrow[dd] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] \arrow[d, equal] & \cdot \arrow[dd] \\
 & \cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] & \\
\cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

is performed by replacing the two middle squares

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] \\
\cdot \arrow[r] & \cdot \\
\cdot \arrow[u] \arrow[r] & \cdot \arrow[u, equal]
\end{tikzcd}$$

by a configuration

$$\begin{tikzcd}
\cdot \arrow[r] & \cdot \\
\cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[u, equal] \arrow[d] \\
\cdot \arrow[r] & \cdot
\end{tikzcd}$$

and then composing vertically. The replacement is a simple pullback construction, and checking that the composed diagram has the same extension as the vertical pasting of the extensions is a straightforward calculation.

11.1.12 The bifibration structure. At the level of polynomials, the bifibration $\textbf{\textit{Poly}}_{\square} \to \textbf{\textit{Set}} \times \textbf{\textit{Set}}$ is now the 'endpoints' functor, associating to a polynomial $I \leftarrow B \to A \to J$ the pair (I, J). With notation as in the proof of Proposition 11.1.7, we know the cobase change of P' along (u, v) is just $v_! \circ P' \circ u^*$, and it is easy to see that We can choose the cotransporter lift of (u, v) to P' to be

$$\begin{tikzcd}
P' : & I' \arrow[d, "u"'] & \cdot \arrow[l] \arrow[r] \arrow[d, equal] & \cdot \arrow[r] \arrow[d, equal] & J' \arrow[d, "v"] \\
(u,v)_{\sharp}(P') : & I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd}$$

so the cobase change of P' is just $v_! \circ P' \circ u^*$. More intrinsically, we can characterise the cotransporter morphisms as the diagrams for which the middle vertical maps are invertible.

The transporter lift of (u, v) to P, which is the same thing as the base change of P along (u, v), is slightly more complicated to construct. It can be given by first base changing along $(u, \mathrm{id}\,)$ and then along $(\mathrm{id}\,, v)$:

$$\begin{tikzcd}
(u,v)^{\sharp}(P) : & I' \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & J' \arrow[d, "v"] \\
(u,\mathrm{id}\,)^{\sharp}(P) : & \cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[dd] & \cdot \arrow[dd, equal] \\
 & \cdot \arrow[d, "u"'] & \cdot \arrow[l] \arrow[d] \arrow[dl, phantom, "\llcorner", very near start] & & \\
P : & I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd}$$

or by first base changing along $(\mathrm{id}\,, v)$ and then along $(u, \mathrm{id}\,)$:

$$\begin{tikzcd}
(u,v)^\sharp(P): & I' \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[dd] & J' \arrow[dd, equal] \\
 & \cdot \arrow[d, "u"'] & \cdot \arrow[l] \arrow[d] \arrow[dl, phantom, "\llcorner", very near start] & & \\
(\mathrm{id}\,, v)^\sharp(P): & \cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d, "v"] \\
P: & I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd}$$

In either case, it can be checked directly that these constructions are indeed the transporter squares. We omit the details.

sourcelift

11.1.13 Sourcelift. The intermediate polynomial $(u, \mathsf{Id})^\sharp(P)$ is called the *source lift* of P along u, and we shall need it later on. Since ∂_0 (as well as ∂_1) is itself a bifibration, for which the source lift is the transporter lift, it enjoys the following universal property: every square

$$\begin{tikzcd}
P': & I' \arrow[dd, "u"'] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] \arrow[d, equal] & J' \arrow[dd, "v"] \\
 & & \cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] & \\
P: & I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd}$$

factors uniquely through the source lift, like

$$\begin{tikzcd}
P': & I' \arrow[dd, equal] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] \arrow[d, equal] & J' \arrow[dd, "v"] \\
 & & \cdot \arrow[u] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] & \\
(u, \mathrm{id}\,)^\sharp(P): & I' \arrow[d, "u"'] & \cdot \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] & \cdot \arrow[d, equal] \\
P: & I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd}$$

where the bottom part is as in (**??**).

Lifts

For each arrow $\alpha : I \to J$ we have the polynomial functor $\alpha_!$, together with canonical diagrams

$$\begin{tikzcd}
\cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[d, "\alpha"] \\
\cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, "\alpha"'] & \cdot
\end{tikzcd}$$

$$\begin{tikzcd}
\cdot \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[r, "\alpha"] \arrow[d, "\alpha"'] & \cdot \arrow[d, equal] \\
\cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot
\end{tikzcd}$$

which are respectively, an cotransporter and a transporter lift to the identity functor.

All this amount to say that α and $\alpha_!$ form a companion pair, in the terminology of Grandis-Paré [45], and the fact that every arrow α has a companion amounts to saying that the double category has a connection, in the terminology of Spencer [98] and Brown-Mosa [22]. Or that it has a folding, in the terminology of Fiore [36].

On the other hand, the assignment of the polynomial functor α^* to an arrow α, and the 2-cells

$$\begin{tikzcd}
\cdot \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[d, equal] \\
\cdot & \cdot \arrow[l, "\alpha"] \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot
\end{tikzcd}$$

$$\begin{tikzcd}
\cdot \arrow[d, equal] & \cdot \arrow[l, "\alpha"'] \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[d, "\alpha"] \\
\cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot
\end{tikzcd}$$

which are a cotransporter and a transporter lift to the identity. This says that every vertical arrow has an orthogonal adjoint (Grandis-Paré).

The four 2-cells indicated constitute in a sense the generic cotransporter-vertical and vertical-transporter factorisations, in the sense that the factorisation of any 2-cell is given by horizontal composition with those cells.

Precisely, the factorisation of a 2-cell $A : P \to Q$ with end components α:

$$\begin{tikzcd}
\cdot \arrow[d, "\alpha"'] \arrow[rrrd, phantom, "A"] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[d, "\alpha"] \\
\cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

into cotransporter followed by vertical is given by:

$$\begin{tikzcd}
\cdot \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[d, equal] \arrow[rrrd, phantom, "\mathsf{Id}"] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[d, "\alpha"] \\
\cdot \arrow[d, equal] & \cdot \arrow[l, "\alpha"'] \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[d, "\alpha"'] \arrow[rrrd, phantom, "A"] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[r, "\alpha"] \arrow[d, "\alpha"] & \cdot \arrow[d, equal] \\
\cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot
\end{tikzcd}$$

Note that the midway polynomial functor, which we denote by $\alpha_\sharp P$, is here on normal form.

And the factorisation of $A : P \to Q$ into vertical followed by transporter is given by:

$$\begin{tikzcd}
\cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[d, "\alpha"'] \arrow[rrrd, phantom, "A"] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[d, "\alpha"] \arrow[r, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[r, equal] \arrow[d, equal] & \cdot \arrow[d, equal] \\
\cdot \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[r, equal] \arrow[d, "\alpha"'] & \cdot \arrow[r, "\alpha"'] \arrow[d, "\alpha"'] & \cdot \arrow[d, equal] \arrow[rrrd, phantom, "\mathsf{Id}"] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[d, equal] & \cdot \arrow[l, "\alpha"] \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[r, equal] \arrow[d, "\alpha"] & \cdot \arrow[d, "\alpha"] \\
\cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot \arrow[r, equal] & \cdot
\end{tikzcd}$$

Note that the midway functor, $\alpha^\sharp Q$, is not on normal form here.

Observation: we have shown (using just formal properties of adjoint, together with one mate) that

$$\alpha_\sharp \dashv \alpha^\sharp$$

It is now a easy verification that these two reindexing functors satisfy the Beck-Chevalley condition.

old calculations

11.1.14 Base change. Given a polynomial functor P represented by $I \leftarrow E \to B \to J$ and set maps $\alpha : \overline{I} \to I$ and $\beta : \overline{J} \to J$, there is a base changed polynomial functor $\overline{I} \leftarrow \overline{E} \to \overline{B} \to \overline{J}$ denoted $(\alpha, \beta)^\sharp P$ with the property that morphisms to P lying over (α, β) are in 1–1 correspondence with natural transformations to $(\alpha, \beta)^\sharp P$. This is to say that any map $Q \to P$ with ends α and β factors uniquely through $(\alpha, \beta)^\sharp P$. All this amounts to saying that the functor $\textbf{\textit{Poly}} \to \textbf{\textit{Set}} \times \textbf{\textit{Set}}$ that returns the endpoints is a Grothendieck fibration.

11.1.15 More details. Given $\alpha : \overline{I} \to I$ and $\beta : \overline{J} \to J$. The formula for $(\alpha, \beta)^\sharp P : \textbf{\textit{Set}}/\overline{I} \to \textbf{\textit{Set}}/\overline{J}$ is easy:

$$(\alpha, \beta)^\sharp P = \alpha_! \cdot P \cdot \beta^*.$$

The is a composite of three polynomial functors:

$$\begin{tikzcd}
& \overline{I} \arrow[r, "="] \arrow[dl, "="'] & \overline{I} \arrow[dr, "\alpha"'] & & E \arrow[r] \arrow[dl] & B \arrow[dr] & & \overline{J} \arrow[r, "="] \arrow[dl, "\beta"] & \overline{J} \arrow[dr, "="] & \\
\overline{I} & & & I & & & J & & & \overline{J}
\end{tikzcd}$$

What is less obvious is that $(\alpha, \beta)^\sharp P$ has a canonical 2-cell to P, and that this 2-cell has the 'transporter' property for the projection functor $\textbf{\textit{Poly}} \to \textbf{\textit{Set}} \times \textbf{\textit{Set}}$. We show that below.

11.1.16 Graphical explanation of base change. It is easy to explain in graphical terms what the base change amounts to and why it has a universal property. If $I \leftarrow E \to B \to J$ is a polynomial functor, then the set of operations B is pictured as the set of bouquets

PICTURE

where each leaf has a decoration in I and the root has one in J. For the polynomial functor pulled back along $\alpha : \overline{I} \to I$ and $\beta : \overline{J} \to J$, the set of operations is the same but where each leaf is also decorated with an element in $\overline{I}$ (required α-compatible with the decoration in I), and the root has also a $\overline{J}$-decoration (β-compatible with the J-decoration). In other words, instead of having just the decoration in I and J imposed by the polynomial functor (i.e. i decorates leaf e if and only if $s(e) = i$ and j decorates the root edge of b if and only if $t(b) = j$) we take all decorations in the α-fibre over $s(e)$ for each leaf e, and the whole β-fibre over $t(b)$ for the root.

$$(\alpha, \beta)^\sharp B = \{B - bouquetswitheachleafdecoratedin\overline{I}suchthat$$

I.e. maps $E_b \to \overline{I}$ over I. And also an element $x \in \overline{J}$ such that $\beta(x) = t(b)$

We call this set $\overline{B}$. The map from $\overline{B} \to B$ just forgets the extra decorations. If $D \leftarrow U \to V \to C$ is another polynomial functor with a map to P in the sense described, it means that each operation in V is mapped to an operation in B in a way compatible with fibres and decorations. It is clear that this map factors through $\overline{B}$ in a unique way compatible with fibres and decorations.

The new decoration on the root edge is just a question of taking a pullback. For the new decoration on the leaves, it amounts to evaluating the polynomial functor on the family $\alpha : \overline{I} \to I$ (except for the last lowershriek). It is intuitively clear that these two operations commute.

Let us now turn to the formal description.

11.1.17 Base change at codomain. Given

$$\begin{tikzcd}
& & & \overline{J} \arrow[d, "\beta"] \\
I & E \arrow[l, "s"'] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

the construction is simply

$$\begin{tikzcd}
I \arrow[d, equal] & E \times_J \overline{J} \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & B \times_J \overline{J} \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{J} \arrow[d, "\beta"] \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

This defines the polynomial functor $(I, \beta)^{\sharp}P = P \cdot \beta^*$ and a map to P.

Note that we think of P as a point of ***Poly*** lying over (I, J), and that we are constructing a transporter lift of the arrow (I, β).

11.1.18 Lemma. *These arrows $(I, \beta)^{\sharp}P$ are strongly transporter.*

THIS PROOF IS ONLY FOR THE CARTESIAN FRAGMENT!

This means that in each diagram, the dashed arrows are unique.

ROTATE DOWN THIS DIAGRAM:

$$\begin{tikzcd}
D \arrow[dr] \arrow[drr, bend left] & & \\
& I \arrow[r, equal] & I \\
U \arrow[uu] \arrow[dd] \arrow[dr, dashed] \arrow[drr, bend left] \arrow[dr, phantom, very near start, "\lrcorner"] & & \\
& E \times_J \overline{J} \arrow[uu] \arrow[r] \arrow[dd] \arrow[ddr, phantom, "\lrcorner"] & E \arrow[uu, "s"'] \arrow[dd, "p"] \\
V \arrow[dd] \arrow[dr, dashed] \arrow[drr, bend left] & & \\
& B \times_J \overline{J} \arrow[r] \arrow[dd] \arrow[ddr, phantom, "\lrcorner"] & B \arrow[dd, "t"] \\
C \arrow[dr] \arrow[drr, bend left] & & \\
& \overline{J} \arrow[r, "\beta"'] & J
\end{tikzcd}$$

This is evident from the universal property of the pullback.

11.1.19 Base change at domain. This is trickier. Given a diagram

$$\begin{tikzcd}
\overline{I} \arrow[d, "\alpha"'] & & & \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

i.e. an arrow (α, J) in ***Set*** $\times$ ***Set*** and an object P in ***Poly*** lying over the codomain, the transporter lift of (α, J) to $I \leftarrow E \rightarrow B \rightarrow J$ is:

$$\begin{tikzcd}
\overline{I} \arrow[d, equal] & p^*p_*s^*\overline{I} \arrow[l] \arrow[r] \arrow[d, "\varepsilon"'] \arrow[dr, phantom, "\lrcorner", very near start] & p_*s^*\overline{I} \arrow[r] \arrow[dd] & J \arrow[dd, equal] \\
\overline{I} \arrow[d, "\alpha"'] & s^*\overline{I} \arrow[l] \arrow[d] \arrow[dl, phantom, "\llcorner", very near start] & & \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

This diagram is clearly nothing but $\alpha_! \cdot P$ (according for the description of composition of polynomial functors), as required.

Note that $s^*\overline{I}$ is the set of B-bouquets with a marked leaf e and with e decorated by an element $i \in \overline{I}$ such that $\alpha(i) = s(e)$. In contrast, $p^*p_*s^*\overline{I}$ is the set of B-bouquets with a marked leaf, but where all leaves has a compatible $\overline{I}$-decoration. The counit ε just forgets all the decorations except at the marked leaf.

11.1.20 Lemma. *These arrows $(I, \beta)^\sharp P$ are strongly transporter.*

THE PROOF IS ONLY ABOUT THE CARTESIAN FRAGMENT

This means that in each diagram, the dashed arrows are unique: TURN DOWN THIS DIAGRAM

$$\begin{tikzcd}
D \arrow[dr] \arrow[drr, bend left] & & \\
& \overline{I} \arrow[r, "\alpha"] & I \\
U \arrow[uu] \arrow[dd] \arrow[dr, dashed] \arrow[drr, bend left] \arrow[dr, phantom, "\lrcorner", very near start] & & \\
& p^*p_*s^*\overline{I} \arrow[uu] \arrow[r] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[uu, "s"'] \arrow[dd, "p"] \\
V \arrow[dd] \arrow[dr, dashed] \arrow[drr, bend left] & & \\
& p_*s^*\overline{I} \arrow[r] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[dd, "t"] \\
C \arrow[dr] \arrow[drr, bend left] & & \\
& J \arrow[r, equal] & J
\end{tikzcd}$$

To see this, observe first that the universal property of the pullback gives a unique map $U \rightarrow s^*\overline{I}$. Since $U = p^*V$, this map corresponds under adjointness $p^* \dashv p_*$ to a map $V \rightarrow p_*s^*\overline{I}$, and the pullback of this map gives us $U \rightarrow p^*p_*s^*\overline{I}$ (which continues to $s^*\overline{I}$

by the counit of the adjointness, and hence makes it fit into the diagram as required).

$$\begin{tikzcd}
D & U \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & V \arrow[r] \arrow[d] & C \arrow[d] \\
\overline{I} \arrow[d, equal] & p^*p_*s^*\overline{I} \arrow[l] \arrow[r] \arrow[d, "\varepsilon"'] \arrow[dr, phantom, very near start, "\lrcorner"] & p_*s^*\overline{I} \arrow[r] \arrow[dd] & J \arrow[dd, equal] \\
\overline{I} \arrow[d, "\alpha"'] & s^*\overline{I} \arrow[l] \arrow[d] \arrow[dl, phantom, very near start, "\llcorner"] & & \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

This constructs the map. Uniqueness follows because different choices of $U \to p_*s^*\overline{I}$ correspond to different choices of $V \to s^*\overline{I}$, but only one is the pullback filler that started the construction.

More precisely we have this:

11.1.21 Lemma. *Given a diagram*

$$\begin{tikzcd}
U \arrow[r] \arrow[d, "w"'] \arrow[dr, phantom, very near start, "\lrcorner"] & V \arrow[dd] \\
X \arrow[d] & \\
E \arrow[r, "p"'] & B,
\end{tikzcd}$$

then the map $\varphi : V \to p_*X$ that corresponds to w under the adjointness $p^* \dashv p_*$ is unique to make this diagram commute:

$$\begin{tikzcd}
U \arrow[r] \arrow[d, "p^*\varphi"'] \arrow[dr, phantom, very near start, "\lrcorner"] \arrow[dd, bend right=60, "w"'] & V \arrow[d, "\varphi"] \\
p^*p_*X \arrow[r] \arrow[d, "\varepsilon"'] \arrow[dr, phantom, very near start, "\lrcorner"] & p_*X \arrow[dd] \\
X \arrow[d] & \\
E \arrow[r, "p"'] & B,
\end{tikzcd}$$

Proof. Suppose a different map φ' could fill the place of φ. Then since the top cartesian square is cartesian over $E \to B$ in the sense that both sides have cartesian squares with $E \to B$, we would have to replace the pullback arrow $p^*\varphi$ by $p^*\varphi'$. But by adjointness bijection, if $\varphi' \neq \varphi$ then also $\varepsilon \circ p^*\varphi' \neq \varepsilon \circ p^*\varphi = w$, so the diagram would not commute in case we change φ. □

Now it is a general fact that composites of strongly transporter maps are again strongly transporter. Hence, since obviously every map in ***Set*** × ***Set*** factors as $(1, \beta)$ followed by $(\alpha, 1)$ (or vice versa), we see that every map has a transporter lift to any target.

Hence we have shown that ***Poly*** $\to$ ***Set*** × ***Set*** is a Grothendieck fibration. Let us be explicit about the general form of a transporter arrow and the general transporter lift: given a diagram

$$\begin{tikzcd}
\overline{I} \arrow[d, "\alpha"'] & & & \overline{J} \arrow[d, "\beta"] \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

the transporter lift of (α, β) to P is given by either of the two diagrams:

$$\begin{tikzcd}
\overline{I} \arrow[d, equal] & p^*p_*s^*\overline{I} \times_J \overline{J} \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & p_*s^*\overline{I} \times_J \overline{J} \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{J} \arrow[d, "\beta"] \\
\overline{I} \arrow[r] \arrow[d, "\alpha"'] & p^*p_*s^*\overline{I} \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & p_*s^*\overline{I} \arrow[r] \arrow[d] & J \arrow[d, equal] \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

$$\begin{tikzcd}
\overline{I} \arrow[d, "\alpha"'] & \overline{p}^*\overline{p}_*\overline{s}^*\overline{I} \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{p}_*\overline{s}^*\overline{I} \arrow[r] \arrow[d] & \overline{J} \arrow[d, equal] \\
I \arrow[r, "\overline{s}"'] \arrow[d, equal] & E \times_J \overline{J} \arrow[r, "\overline{p}"'] \arrow[d, "a"] \arrow[dr, phantom, very near start, "\lrcorner"] & B \times_J \overline{J} \arrow[r] \arrow[d, "b"] \arrow[dr, phantom, very near start, "\lrcorner"] & \overline{J} \arrow[d, "\beta"] \\
I & E \arrow[l, "s"] \arrow[r, "p"'] & B \arrow[r, "t"'] & J
\end{tikzcd}$$

We shall also call it the pullback of P along (α, β).

There is of course a canonical identification of the top rows: This follows from the general factorisation result. It also follows from associativity of composition of polynomial functors. But let us provide the comparison explicitly: we need a map (canonical and unique, but since this follows from the general result, we just care to exhibit the map):

$$\overline{p}_*\overline{s}^*\overline{I} \overset{\sim}{\to} (p_*s^*\overline{I}) \times_J \overline{J}.$$

Since we already have a map $\overline{p}_*\overline{s}^*\overline{I} \to \overline{J}$ we just need to exhibit a map to $p_*s^*\overline{I}$ (and the two maps should agree down in J). The source of our map is really $b_!\,\overline{p}_*\overline{s}^*\overline{I}$. Note that $\overline{s} = s \circ a$. Now:

$$b_!\,\overline{p}_*\overline{s}^*\overline{I} = b_!\,\overline{p}_*a^*s^*\overline{I} \to p_*a_!\,a^*s^*\overline{I} \to p_*s^*\overline{I}$$

The first arrow is the one of Lemma 8.3.2, the second arrow is the counit for $a_! \dashv a^*$.

11.1.22 Example. If we restrict to the category of linear polynomial functors—these are just spans, then the factorisation reduces to the well-known fact that a map of spans

$$\begin{tikzcd}
D \arrow[d] & M \arrow[l] \arrow[r] \arrow[d] & C \arrow[d] \\
I & B \arrow[l] \arrow[r] & J
\end{tikzcd}$$

factors uniquely through $D \leftarrow D \times_I B \times_J C$.

ATTENTION! THERE MIGHT BE MORE 2-CELLS FOR LINEAR FUNCTORS THAN FOR SPANS, BECAUSE WE MIGHT HAVE EXTRA BIJECTIONS IN THE MIDDLE!

PROJECT TO INVESTIGATE MORE CLOSELY. There is probably a large class of weak double categories with this features: the category of horizontal arrows and 2-cells is fibred over the vertical category. This means that every square factors (vertically) as 'essential' followed by 'transporter'. Here 'essential' is supposed to mean with trivial vertical arrows as endpoints, and 'transporter' is supposed to mean that there is a 'base-change' of the bottom horizontal arrow along the two vertical ones. . .

This should be true for module categories: to give a square in the double category of modules

$$\begin{tikzcd}
R \arrow[r, "M"] \arrow[d, "f"'] & S \arrow[d, "g"] \\
R' \arrow[r, "N"'] & S'
\end{tikzcd}$$

is to give an (f,g)-equivariant module map $M \to N$, which in turn should be equivalent to giving a R-S-module map $M \to (f,g)^\sharp N$, with suitable better notation.

CHECK OUT PROFUNCTORS AND SPANS

11.2 Horizontal composition

11.2.1 Horizontal composition of 2-cells. Let us indicate how to compute a horizontal composition at the intensional level. After inserting a few identity arrows, the diagram for a general horizontal composition in *BRDG* is this:

$$\begin{tikzcd}
I' & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J' & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & K' \\
\cdot \arrow[u, equal] \arrow[d] & \cdot \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[u, equal] \arrow[d] & \cdot \arrow[u, equal] \arrow[d] & \cdot \arrow[l] \arrow[r] \arrow[u] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[u, equal] \arrow[d] & \cdot \arrow[u, equal] \arrow[d] \\
I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & K
\end{tikzcd}$$

Invoking the interchange law, we can perform this composition row-wise: the top strip is the horizontal composition of two 'representable' natural transformations. We now proceed to describe this. The bottom strip is a horizontal composition entirely within the cartesian fragment, which we describe in the next subsection.

We shall see that it is not true that the class of cartesian (base changes) is stable. We shall exhibit very explicitly what goes wrong and what works well.

Given a horizontal composite of base-change squares

$$\begin{tikzcd}
\cdot \arrow[d] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[d] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[d] \\
\cdot & \cdot \arrow[l] \arrow[r] \arrow[u, phantom, "P"] & \cdot \arrow[r] & \cdot & \cdot \arrow[l] \arrow[r] \arrow[u, phantom, "Q"] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

we factor P vertically as source-lift followed by a target-lift, and we factor Q the other way around:

$$\begin{tikzcd}
\cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \arrow[d] & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot \\
\cdot \arrow[u, equal] \arrow[d] & \cdot \arrow[l] \arrow[r] \arrow[u, dash] \arrow[d, dash] \arrow[ur, phantom, "\lrcorner", very near end] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[u, dash] \arrow[d, dash] \arrow[ur, phantom, "\lrcorner", very near end] & \cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[u, dash] \arrow[d, dash] \arrow[ur, phantom, "\lrcorner", very near end] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[u, dash] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[u, equal] \arrow[d] \\
\cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

Now we invoke the interchange law: we can compose the diagram by composing each row of 2-cells horizontally.

The top row of 2-cells is clearly vertical for the Grothendieck fibration (and it is clear that in general it is not the vertical identity 2-cell). Hence we see already that the horizontal composite of two base-change squares is not a base-change square.

Let us show that the horizontal composite of the bottom row of 2-cells is again a base-change 2-cell:

11.2.2 Lemma. *A target-lift* 2*-cell horizontally followed by a source-lift* 2*-cell is again a base-change cell.*

Proof. We take advantage of the interchange law once again (and the fact that the gluing locus is a vertical identity 1-cell) to insert two vertical identity 2-cells break the proof into two pieces, each of independent interest. Namely write the composite

$$\begin{tikzcd}
\cdot \arrow[d] & \cdot \arrow[l] \arrow[r] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d, dash] & \cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] \\
\cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

as

$$\begin{tikzcd}
\cdot \arrow[d] & \cdot \arrow[l] \arrow[r] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d, dash] & \cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d, equal] & \cdot \arrow[r] \arrow[d, equal] & \cdot \arrow[d, equal] \\
\cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d, equal] & \cdot \arrow[r] \arrow[d, equal] & \cdot \arrow[d, equal] & \cdot \arrow[l] \arrow[r] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d, dash] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[d] \\
\cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & \cdot
\end{tikzcd}$$

Now the top row horizontal composite is a source-lift and the bottom row horizontal composite is a target lift, according to the following two lemmas.

□

11.2.3 Lemma. *A source-lift* 2*-cell post-whiskered by a* 1*-cell is again a source-lift* 2*-cell.*

Proof. □

11.2.4 Lemma. *A target-lift* 2*-cell pre-whiskered by a* 1*-cell is again a target-lift* 2*-cell.*

Proof. □

11.3 Cartesian

11.3.1 Cartesian fragment. All the constructions and arguments of this section apply equally well to the cartesian case: start with the 2-category of polynomial functors and their cartesian fibred natural transformations, and construct a double category out of it. Its squares are cartesian (fibred) natural transformations

$$\begin{tikzcd}
\boldsymbol{Set}/I' \arrow[r, "P'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\swarrow"] & \boldsymbol{Set}/J' \arrow[d, "v_!"] \\
\boldsymbol{Set}/I \arrow[r, "P"'] & \boldsymbol{Set}/J
\end{tikzcd} \tag{11.7}$$

The corresponding diagrams forming the 2-cells at the intensional level are simply

$$\begin{tikzcd}
I' \arrow[d, "u"'] & \cdot \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \cdot \arrow[r] \arrow[d] & J' \arrow[d, "v"] \\
I & \cdot \arrow[l] \arrow[r] & \cdot \arrow[r] & J
\end{tikzcd} \tag{11.8}$$

cartesian2cell

All the constructions are compatible with the cartesian condition, since they all depend on the lowershriek-upperstar adjunction, which is cartesian. Note also that the transporter and cotransporter lifts belong to the cartesian fragment.

The following two results follow readily.

11.3.2 Proposition. *The double category* $\boldsymbol{PolyFun}^{c}_{\square}$ *whose objects are the slices of* ***Set****, whose horizontal arrows are the polynomial functors, whose vertical arrows are the dependent sum functors, and whose squares are cartesian strong natural transformations*

$$\begin{tikzcd}
\boldsymbol{Set}/I' \arrow[r, "P'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\swarrow"] & \boldsymbol{Set}/J' \arrow[d, "v_!"] \\
\boldsymbol{Set}/I \arrow[r, "P"'] & \boldsymbol{Set}/J
\end{tikzcd}$$

is a framed bicategory.

11.3.3 Proposition. *The squares of* $\boldsymbol{PolyFun}^c_\square$ *are represented uniquely by diagrams* (11.8). *Hence extension constitutes a framed biequivalence*

$$\boldsymbol{Poly}^c_\square \overset{\sim}{\to} \boldsymbol{PolyFun}^c_\square .$$

The cartesian fragment is in many respect more well-behaved than the full double category. For example, horizontal composition of 2-cells like (11.8) is essentially straightforward, as we shall now see.

Horizontal composition of cartesian 2-cells

It is just a question of building the composites of the top and bottom polynomial functors as in **??** and observing that the connecting arrows can be carried along in this construction (either by naturality or by the pullback property).

The vertical arrows are just set maps (or more precisely, pullback maps or lowershriek maps between the corresponding slice categories. The horizontal arrows are polynomial functors. The 2-cells are diagrams

$$\begin{tikzcd}
I' \arrow[d] & E' \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B' \arrow[r] \arrow[d] & J' \arrow[d] \\
I & E \arrow[l] \arrow[r] & B \arrow[r] & J
\end{tikzcd}$$

It is clear how to compose vertically: this is just to compose arrows and compose such 2-cells vertically, and since the stacking of two pullback squares is again a pullback square, this gives again a 2-cell.

For horizontal composition, we have already shown how to compose polynomial functors to get new polynomial functors, and we also know how to compose cartesian natural transformations horizontally to get a new cartesian natural transformation.

Now we need to compose more general things, and these are not natural transformations in any category—at least not apparently.

$$\begin{tikzcd}
I' \arrow[d] & E' \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B' \arrow[r] \arrow[d] & J' \arrow[d] & F' \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & C' \arrow[r] \arrow[d] & K' \arrow[d] \\
I & E \arrow[l] \arrow[r] & B \arrow[r] & J & F \arrow[l] \arrow[r] & C \arrow[r] & K
\end{tikzcd}$$

What we need to do is to take two parallel big composition diagrams like (8.7), place them side by side, and fill in maps between their bases, making a commutative base with cartesian squares in the middles. Now we must show that the sides of this construction yields new connecting arrows, and that the roof of the total construction is a pair of cartesian squares.

It is quite easy to check that the left-hand part of the diagram gives a cartesian square on the top: it has a cartesian square as basis, and the sides are all pullbacks. Hence the top is a pullback. For the right-hand part of the diagram a few less obvious arguments are needed.

Step 1: in the middle of the basis we have

$$\begin{tikzcd}
 & & & W' \arrow[dll, dashed] \arrow[dl] \arrow[dr] & \\
 & W \arrow[dl] \arrow[dr] & B' \arrow[dll] \arrow[dr] & & F' \arrow[dll] \arrow[dl] \\
B \arrow[dr] & & F \arrow[dl] & J' \arrow[dll] & \\
 & J & & &
\end{tikzcd}$$

Front and back squares are pullbacks, so the dashed arrows exists by the universal property of the front pullback.

Step 2: take push-forth of W and W' along q and q', respectively. We want to see that there is a natural map from q'_*W to q_*W. This map will be the crucial top-right edge of the final roof.

Let c' be an element in C' and let c be its image in C under the map we were given $C' \to C$. Now we know that the fibre over c' of q'_*W' is the set of diagrams

$$\begin{tikzcd}
 & & W' \arrow[d] \\
F'_{c'} \arrow[urr, "s"] & \subset & F'
\end{tikzcd}$$

Given one such diagram (i.e. an element in the set $(q'_*W')_{c'}$) we want to construct a diagram

$$\begin{tikzcd}
 & W \arrow[d] \\
F_c \arrow[ur, dashed] \arrow[r, phantom, "\subset"] & F
\end{tikzcd}$$

i.e. an element in $(q_* W)_c$ (There are of course some compatibility conditions to check.) But since the square

$$\begin{tikzcd}
F' \arrow[r] \arrow[d] & C' \arrow[d] \\
F \arrow[r] & C
\end{tikzcd}$$

is a pullback we have a bijection $F'_{c'} \xrightarrow{\sim} F_c$, so the dashed arrow we want is the one arising naturally in this diagram:

$$\begin{tikzcd}
 & W' \arrow[dd] \arrow[dr] & \\
F'_{c'} \arrow[ur] \arrow[rr, hook] \arrow[dd, equal] & & F' \arrow[dd] \\
 & W \arrow[dr] & \\
F_c \arrow[ur, dashed] \arrow[rr, hook] & & F
\end{tikzcd}$$

Step 3: now pull back those push-forths. We get the top edges of the total roof. Since the walls are pullbacks, and the bottom is, then so is the top. . .

Misc issues in the cartesian fragment

Surjection-injection factorisation in *Poly*

A cartesian morphism α of polynomial functors is called *injective*, resp. *surjective*, if each of the three components, $\alpha^0, \alpha^1, \alpha^2$ is injective, resp. surjective.

fact-Poly **11.3.4 Lemma.** *Every arrow in* ***Poly*** *factors uniquely as one where all the components are epi followed by one where all the components are mono.*

(I don't know if these are the categorical notions of epi and mono.)

Proof. This is straightforward: given

$$\begin{tikzcd}
I' \arrow[r] \arrow[d] & E' \arrow[r] \arrow[d] & B' \arrow[r] \arrow[d] & I' \arrow[d] \\
I'' \arrow[r] & E'' \arrow[r] & B'' \arrow[r] & I''
\end{tikzcd}$$

factor each of the vertical maps into epi-mono:

$$\begin{tikzcd}
I' \arrow[r] \arrow[d, two heads] & E' \arrow[r] \arrow[d, two heads] & B' \arrow[r] \arrow[d, two heads] & I' \arrow[d, two heads] \\
I \arrow[r] \arrow[d, tail] & E \arrow[r] \arrow[d, tail] & B \arrow[r] \arrow[d, tail] & I \arrow[d, tail] \\
I'' \arrow[r] & E'' \arrow[r] & B'' \arrow[r] & I''.
\end{tikzcd}$$

Here you check easily that the middle squares are again cartesian: indeed if you take the pullback in the bottom

$$\begin{tikzcd}
P \arrow[r] \arrow[d, tail] & B \arrow[d, tail] \\
E'' \arrow[r] & B''
\end{tikzcd}$$

you get a mono (since a pullback of a mono is always a mono). Next by the universal property of this pullback, there is a map from E' to P, and hence the top square is a pullback too. Now in the category of sets it is also true that the pullback of an epi is an epi, so by uniqueness of epi-mono factorisation in ***Set***, $P = E$. □

Sums and products in the variable-type categories

11.3.5 Sums. Let's try:

$$I_1 \leftarrow E_1 \rightarrow B_1 \rightarrow J_1$$

$$+$$

$$I_2 \leftarrow E_2 \rightarrow B_2 \rightarrow J_2$$

$$:=$$

$$I_1 + I_2 \leftarrow E_1 + E_2 \rightarrow B_1 + B_2 \rightarrow J_1 + J_2$$

It is easy to see that the sum injections are cartesian. This follows from the observation that this square is cartesian

$$\begin{tikzcd}
E_1 \arrow[r] \arrow[d] & B_1 \arrow[d] \\
E_1 + E_2 \arrow[r] & B_1 + B_2
\end{tikzcd}$$

Universal property: given $Q : \ D \leftarrow U \rightarrow V \rightarrow C$ with cartesian maps from P_1 and P_2, then there is induced a unique map $P_1 + P_2 \Rightarrow Q$.

11.3.6 Products.

Coherence problems

We know how to compose bridges, and we know that this composition law corresponds to composition of the associated functors. We have not justified how to horizontally compose 2-cells. And we have not proved that the resulting 2-cell between the composites induces the pasted 2-cell in the category of functors.

Here is more formally the coherence problem: given two composable bridges P and Q, we have constructed a composite bridge which we denote $Q \circ P$. It is connected to the two original bridges by three pullback squares and one pentagon. We have shown that this big diagram induces a natural isomorphism

$$\widetilde{Q} \circ \widetilde{P} \overset{\Phi_{P,Q}}{\Leftrightarrow} \widetilde{Q \circ P}.$$

This isomorphism is a composite of three Beck-Chevalley isomorphisms and one distributivity isomorphism coming from the pentagon.

Now suppose we have 2-cells from bridge P' to bridge P, and from bridge Q' to bridge Q. We have also shown that these 2-cells induce natural transformations

$$\begin{tikzcd}
\cdot \arrow[r, "P'"] \arrow[d, "!"'] \arrow[dr, phantom, "\swarrow"] & \cdot \arrow[r, "Q'"] \arrow[d, "!"'] \arrow[dr, phantom, "\swarrow"] & \cdot \arrow[d, "!"'] \\
\cdot \arrow[r, "P"'] & \cdot \arrow[r, "Q"'] & \cdot
\end{tikzcd}$$

Our task is first to build a 2-cell from bridge $Q' \circ P'$ to bridge $Q \circ P$, and then to verify this equation of natural transformation of functors:

$$\begin{tikzcd}
\cdot \arrow[rr, "\widetilde{Q' \circ P'}"] \arrow[dr, "\widetilde{P'}"'] \arrow[dd] & \arrow[d, phantom, "\Phi_{P',Q'}"] & \cdot \arrow[dd] & & \cdot \arrow[rr, "\widetilde{Q' \circ P'}"] \arrow[dd] & & \cdot \arrow[dd] \\
& \cdot \arrow[ur, "\widetilde{Q'}"'] \arrow[dd] & & = & & & \\
\cdot \arrow[dr, "\widetilde{P}"'] & & \cdot & & \cdot \arrow[rr, "\widetilde{Q \circ P}"] \arrow[dr, "\widetilde{P}"'] & \arrow[d, phantom, "\Phi_{P,Q}"] & \cdot \\
& \cdot \arrow[ur, "\widetilde{Q}"'] & & & & \cdot \arrow[ur, "\widetilde{Q}"'] &
\end{tikzcd}$$

In the cartesian fragment, it should not be too difficult, although cumbersome perhaps: I think the connecting maps defining the horizontal composite of the squares induce connecting maps between the composites of the bridges.

Chapter 12

Trees (1)

Ch:tEmb

This chapter is mostly copied verbatim from the paper *Polynomial functors and trees* [63].

In this chapter we start the study of the close relationship between polynomial endofunctors and trees. We shall see how a tree gives rise to a polynomial endofunctor, and in fact we characterise those polynomial endofunctors that arise in this way. So in the end we can *define* trees as a special kind of polynomial endofunctors.

One single observation accounts for the close relationship between polynomial endofunctors and trees, namely that they are represented by diagrams of the same shape, as we now proceed to explain. Although this observation is both natural and fruitful, it seems not to have been made before.

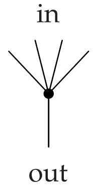
Trees are usually defined and manipulated in either of two ways:

- 'Topological/static characterisation': trees are graphs $E \rightrightarrows V$ with certain topological properties and structure (a base point).

- 'Recursive characterisation': a tree is either a trivial tree or a collection of smaller trees.

Here we take a different viewpoint, specifically designed for the use of trees in operad theory and related topics:

- 'Operational characterisation': trees are certain many-in/one-out structures, i.e. built from building blocks like



Accordingly a tree should have a set of edges A, and a set of vertices (or nodes) N, which we think of as operations; these should have inputs and output (source and target). So the structure is something like

$$\begin{tikzcd}
 & N \arrow[dl, "\mathrm{in}"', dash] \arrow[dr, "\mathrm{out}"] & \\
A & & A.
\end{tikzcd}$$

The fork represents a 'multi-valued map', because a node may have several input edges. A standard way to encode multi-valued maps is in terms of correspondences or spans; hence we arrive at this shape of diagram to represent a tree:

$$A \xleftarrow{s} M \xrightarrow{p} N \xrightarrow{t} A; \tag{12.1}$$

diagram

to be explicit, M is the set of all input edges (i.e. pairs (b, e) where b is a node and e is an input edge of b). In order to be trees, such diagrams should satisfy certain axioms, which turn out to be quite simple.

Although this is clearly also a static graph-like definition, its operational aspect comes to the fore with the observation that this shape of diagram is precisely what defines polynomial endofunctors, vindicating the interpretation of N as a set of operations.

The recursive aspect of trees is also prominent in the present approach, remembering that polynomial endofunctors provide categorical semantics for inductive data types (W-types), the latter appearing as initial Lambek algebras for the former [81]. In fact, a recursive characterisation of trees (12.3.21) follows quite easily from the definition. While in type theory trees (of a certain branching profile P) appear as initial algebras *for* some polynomial functor P (expressing the branching profile), in this work trees *are* themselves certain polynomial functors. In a precise sense they are absolute trees, i.e. not relative to any preassigned branching profile.

We define a tree to be a diagram of sets of shape (12.1) satisfying four simple conditions (12.2.3). The category ***TEmb*** is the full subcategory of

PolyEnd (the category of polynomial endofunctors) consisting of the trees. The morphisms are diagrams

$$\begin{tikzcd}
A' \arrow[d] & M' \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & N' \arrow[r] \arrow[d] & A' \arrow[d] \\
A & M \arrow[l] \arrow[r] & N \arrow[r] & A.
\end{tikzcd}$$

The symbol ***TEmb*** stands for 'tree embeddings', as it turns out maps between trees are always injective (12.3.3) and correspond to a notion of subtree. Root-preserving embeddings and ideal embeddings are characterised in terms of pullback conditions, and every tree embedding factors as root-preserving followed by ideal embedding (12.3.15). These two classes of maps allow pushouts along each other in the category ***TEmb***—this is grafting of trees (12.3.19). This leads to a recursive characterisation of trees (12.3.21), as well as the useful result that every tree is the iterated pushout of its one-node subtrees over its inner edges (12.3.24).

For a polynomial endofunctor P, a P-tree is a tree with a map to P. This amounts to structuring the set of input edges of each node. For example, if M is the free-monoid monad (1.2.8), then M-trees are precisely planar trees. The notion of P-tree is crucial for the study of monads, which we take up again in the next Chapter: we shall show, using the recursive characterisation of trees, that the set of isomorphism classes of P-trees, denoted $\mathrm{tr}(\mathsf{P})$, is the least fixpoint (initial Lambek algebra) for the polynomial endofunctor $1 + \mathsf{P}$ (Theorem 12.4.6). This leads to the following explicit construction of the free monad on P: if P is given by the diagram $A \leftarrow M \rightarrow N \rightarrow A$, then the free monad on P is given by

$$A \leftarrow \mathrm{tr}'(\mathsf{P}) \rightarrow \mathrm{tr}(\mathsf{P}) \rightarrow A$$

where $\mathrm{tr}'(\mathsf{P})$ denotes the set of isomorphism classes of P-trees with a marked leaf (14.1.2). The monad structure is given by grafting P-trees.

After studying monads, we shall come back to trees, introducing the category of trees in terms of monads: it is the category of free monads on trees. Since maps between trees are embeddings, the free monad on a tree $\mathsf{T} = (A \leftarrow M \rightarrow N \rightarrow A)$ is given by

$$A \leftarrow \mathrm{sub}'(\mathsf{T}) \rightarrow \mathrm{sub}(\mathsf{T}) \rightarrow A$$

(where $\mathrm{sub}(\mathsf{T})$ (resp. $\mathrm{sub}'(\mathsf{T})$) denotes the set of subtrees of T (resp. subtrees with a marked leaf)). We shall define ***Tree*** to be the category whose objects are trees and whose arrows are maps between their free monads (14.2.1).

12.1 Trees

12.1.1 Graphs. By a *graph* we understand a pair (T_0, T_1), where T_0 is a set, and T_1 is a set of subsets of T_0 of cardinality 2. The elements in T_0 are called *vertices*, and the elements in T_1 *edges*. An edge $\{x, y\}$ is said to be *incident* to a vertex v if $v \in \{x, y\}$. We say a vertex is of valence n if the set of incident edges is of cardinality n.

Say what a map of graphs is, and then what a subgraph is.

Say what a simple path is. Perhaps via maps from a linear graph.

A *simple closed path* in a graph is a subgraph $C \subset T$ with the property that every vertex is incident to precisely two edges. A graph is said to be simply connected if it does not contain a simple closed path.

The geometric realisation of a graph is the CW-complex with a 0-cell for each vertex, and for each edge a 1-cell attached at the points corresponding to its two incident vertices. A graph is connected (resp. simply connected) if and only if its geometric realisation is connected (resp. simply connected).

tree-formal

12.1.2 Trees. By a *finite rooted tree with boundary* we mean a finite graph $T = (T_0, T_1)$, connected and simply connected, equipped with a pointed subset ∂T of vertices of valence 1. We will not need other kinds of tree than finite rooted trees with boundary, and we will simply call them *trees*.

The basepoint $t_0 \in \partial T$ is called the the *output vertex*, and the remaining vertices in ∂T are called *input vertices*. Most of the time we shall not refer to the boundary vertices at all, and graphically a boundary vertex is just represented as a loose end of the incident edge. Edges incident to input vertices are called *input edges* or *leaves*, while the unique edge incident to the output vertex is called the *output edge* or the *root edge*.

The set of vertices $c(T) := T_0 \smallsetminus \partial T$ is called the *core* of the tree, and its elements are called *nodes* or *dots*; we draw them as dots. A tree may have zero dots, in which case it is just a single edge (together with two boundary vertices, which we suppress). We call such a tree a *unit tree*. For a tree with at least one dot, the dot incident to the output edge is called the *root dot*. Not every vertex of valence 1 needs to be a boundary vertex: those which are not are called *nullary dots*.

The standard graphical representation of trees is justified by geometric realisation. Note that leaves and root are realised by half-open intervals, and we keep track of which are which by always drawing the output at the bottom. By labelling the cells we can recover the abstract tree when needed, and we shall allow ourselves to mix the two viewpoints, although we shall frequently omit the labels.

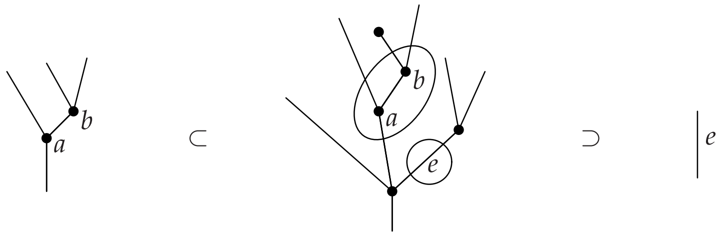
12.1.3 Tree order. If $T = (T_0, T_1, \partial T, t_0)$ is a tree, the set T_0 has a natural poset structure $a \leq b$, in which the input vertices are minimal elements and the output vertex is the maximal element. A poset defined by a tree we call a *tree order*. We say a is a *child* of b if $a \leq b$ and $\{a, b\}$ is an edge.

Since T is simply connected, each edge divides T into two connected components. One component contains the root vertex. This defines an orientation of the edge: we say the end of the edge is the vertex in the component of the root, and the start of the edge is the other vertex of the edge. At every vertex (not the root) there is precisely one edge starting. Indeed, if there were two, each of them would be the first step in a path ending at the root vertex (possibly meeting before), and hence we would have a simple closed path, contradicting the simple connectedness. Hence every dot has an out-edge. The other edges incident to the dot are called input edges. The orientation of the edges assemble together into a partial order on the set of edges. (and also a partial order on the set of vertices).

Need to see that at every dot of a tree, there is precisely one edge going out of it. Any number of edges going into it.

subtree

12.1.4 Subtrees. A *subtree* in T is a subgraph S with a tree structure (in the sense of 12.1.2) such that each inner vertex of S is also an inner vertex of T, and the set of input edges of an inner vertex (i.e. a node) in S coincides with the set of input edges of that node in T. Finally we require compatibility between the tree orders of S and T (this condition is automatically satisfied whenever S contains an inner vertex). Geometrically, if T is embedded in the plane, a subtree is a non-empty connected subgraph that can be cut out by a circle (or simple closed curve) that does not meet the vertices. The boundary of the subtree is then the intersection with the circle. Here are two examples:



Note that each edge of T defines a dotless subtree.

12.2 From trees to polynomial endofunctors

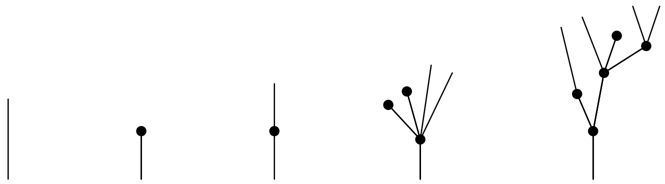
In this chapter we only use endofunctors. Throughout we use sans serif typeface for polynomial endofunctors, writing $\mathsf{P} = (P^0, P^1, P^2)$ for the functor represented by

$$\begin{tikzcd} P^0 & P^2 \arrow[l, "s"'] \arrow[r, "p"] & P^1 \arrow[r, "t"] & P^0. \end{tikzcd}$$

We shall use the letters s, p, t for the three arrows in any diagram representing a polynomial endofunctor.

We shall define trees to be certain polynomial endofunctors. To motivate this definition, let us first informally explain what trees are supposed to be, and then show how to associate a polynomial endofunctor to a tree.

12.2.1 Trees. Our trees are non-planar finite rooted trees with boundary. Each node has a finite number of incoming edges and precisely one outgoing edge, always drawn below the node. The following drawings should suffice to exemplify trees, but beware that the planar aspect inherent in a drawing should be disregarded:



Note that certain edges (the *leaves*) do not start in a node and that one edge (the *root edge*) does not end in a node. The leaves and the root together form the *boundary* of the tree.

We shall give a formal definition of tree in a moment (12.2.3).

polyfromtree

12.2.2 Polynomial functors from trees. Given a tree, define a polynomial functor

$$\begin{tikzcd} T^0 & T^2 \arrow[l, "s"'] \arrow[r, "p"] & T^1 \arrow[r, "t"] & T^0, \end{tikzcd}$$

by letting T^0 be the set of edges, T^1 the set of nodes, and T^2 the set of nodes with a marked input edge, i.e. the set of pairs (b, e) where b is a node and e is an incoming edge of b. The maps are the obvious ones: s returns the marked edge of the node (i.e. $(b, e) \mapsto e$), the map p forgets the mark (i.e. $(b, e) \mapsto b$), and t returns the output edge of the node.

For example, the first three trees in the drawing above correspond to the following polynomial functors:

$$1 \leftarrow 0 \rightarrow 0 \rightarrow 1 \qquad\qquad 1 \leftarrow 0 \rightarrow 1 \rightarrow 1 \qquad\qquad 2 \leftarrow 1 \rightarrow 1 \rightarrow 2.$$

The polynomial functors that arise from this construction are characterised by four simple conditions which are convenient to work with. We shall take this characterisation as our definition of tree:

polytree-def

12.2.3 Definition of tree. We define a *finite rooted tree with boundary* to be a polynomial endofunctor $\mathsf{T} = (T^0, T^1, T^2)$

$$\begin{tikzcd} T^0 & T^2 \arrow[l, "s"'] \arrow[r, "p"] & T^1 \arrow[r, "t"] & T^0 \end{tikzcd}$$

satisfying the following four conditions:

(1) all the involved sets are finite.

(2) t is injective

(3) s is injective with singleton complement (called the *root* and denoted 1).

With $T^0 = 1 + T^2$, define the walk-to-the-root function $\sigma : T^0 \to T^0$ by $1 \mapsto 1$ and $e \mapsto t(p(e))$ for $e \in T^2$.

(4) $\forall x \in T^0 : \exists k \in \mathbb{N} : \sigma^k(x) = 1$.

The elements of T^0 are called *edges*. The elements of T^1 are called *nodes*. For $b \in T^1$, the edge $t(b)$ is called the *output edge* of the node. That t is injective is just to say that each edge is the output edge of at most one node. For $b \in T^1$, the elements of the fibre $(T^2)_b := p^{-1}(b)$ are called *input edges* of b. Hence the whole set $T^2 = \sum_{b \in T^1} (T^2)_b$ can be thought of as the set of nodes-with-a-marked-input-edge, i.e. pairs (b, e) where b is a node and e is an input edge of b. The map s returns the marked edge. Condition (3) says that every edge is the input edge of a unique node, except the root edge. Condition (4) says that if you walk towards the root, in a finite number of steps you arrive there.

The edges not in the image of t are called *leaves*. The root and the leaves together form the *boundary* of the tree.

From now on we just say *tree* for 'finite rooted tree with boundary'.

Let us briefly describe how to draw such a tree, i.e. give the converse of the construction in 12.2.2. Given (T^0, T^1, T^2) we define a finite, oriented graph with boundary, i.e. edges are allowed to have loose ends: take the vertex set to be T^1 and the edge set to be T^0. The edges $x \in T^0$ which are not in the image of t are the input edges of the graph in the sense that they do not start in a vertex. For each other edge x, we let it start in b if and only if $t(b) = x$. (Precisely one such b exists by axiom (2).) Clearly every b occurs like this. Now we have decided where each edge starts. Let us decide where they end: the root edge 1 is defined to be the output edge of the graph, in the sense that it does not end in a vertex. For each other edge $e \neq 1$ (which we think of as $e \in T^2$), we let it end in $p(e)$. Note that the fibre of p over a vertex b consists of precisely the edges ending in b. Now we have described how all the edges and vertices are connected, and hence we have described a finite, oriented graph with boundary. Condition (4) implies that the graph is connected: every $e \neq 1$ has a 'next edge' $\sigma(e)$ distinct from itself, and in a finite number of steps comes down to the root edge. There can be no loops because there is precisely one edge coming out of each vertex, and linear cycles are excluded by connectedness and existence of a root. In conclusion, the graph we have drawn is a tree.

More formal proof: define a directed graph $T = (T_0, T_1)$ by taking $T_1 := I$ and $T_0 := I + 1 = B + L + 1$ (where L denotes the complement of B in I, and 1 denotes the root vertex). Define the startpoint of each edge by setting

$$\text{start} : I \to I + 1$$

and define the endpoint as

$$\text{end} : I = E + 1 \to I + 1$$

defined as $\rho + 1$. Here the 1 on the left denotes the root edge, while the 1 on the right denotes the root vertex.

Observe that start and end are always distinct: for $e \in E$ we certainly have $\rho(e) \neq e$, and for r the root edge in I, then $\text{start}(r) = r$ and $\text{end}(r) = 1$ (the root vertex). We should also check that there are no multiple edges. . .

Connectedness: every edge e has a neighbour $\rho(e)$ or otherwise e is the root edge. The edge $\rho(e)$ again has a lower neighbour, and so on until we reach the root, so every edge is connected to the root.

Simple connectedness: define the height $h(x)$ of a dot x as the minimal number h such that $\rho^h(t(x)) = 1$ (the root edge). Suppose there is a simple closed path in T. Then there is a finite number of vertices involved, so there is a x where $h(x)$ is maximal. But this contradicts the assumption that x has 2 incident edges: at most one of the edges is an output edge, so at least one is an input edge, and this edge has another incident vertex in the loop, hence contradicts the maximality of x.

12.2.4 The trivial tree. The nodeless tree

$$\begin{tikzcd} 1 & 0 \arrow[l] \arrow[r] & 0 \arrow[r] & 1, \end{tikzcd}$$

(consisting solely of one edge) is called the *trivial tree*, and is denoted $\shortmid$.

one-node

12.2.5 One-node trees. For each finite set E we have a *one-node tree*,

$$\begin{tikzcd} E + 1 & E \arrow[l, "s"'] \arrow[r, "p"] & 1 \arrow[r, "t"] & E + 1, \end{tikzcd}$$

where s and t are the sum inclusions.

12.2.6 Elementary trees. An *elementary tree* is one with at most one node. That is, either a trivial tree or a one-node tree. These will play a fundamental role in the theory. We shall see in a moment that every tree is obtained by gluing together one-node trees along trivial trees in a specific way (grafting), while polynomial endofunctors are more general colimits of elementary trees.

12.2.7 Terminology. We define a partial order (called the *tree order*) on the edge set T^0 by declaring $x \leq y$ when $\exists k \in \mathbb{N} : \sigma^k(x) = y$. In this case x is called a *descendant* of y, and y is called an *ancestor* of x. In the particular case where $\sigma(x) = y$ and $x \neq y$, we say that x is a *child* of y. If $\sigma(x) = \sigma(y)$ and $x \neq y$ we say that x and y are *siblings*. We define the *distance* from x to y to be $\min\{k \in \mathbb{N} \mid \sigma^k(x) = y\}$, whenever this set is nonempty. Note that the order induced on any 'upset' is a linear order: if $e \leq x$ and $e \leq y$ then $x \leq y$ or $y \leq x$. The poset T^0 has a maximal element, namely the root; hence it has binary joins: the join of two edges is their nearest common ancestor. Every leaf is a minimal element for the tree order, but there may be other minimal elements. (Note that a partial order is induced on $T^2 \subset T^0$, and also on T^1 (via t).)

Examples of trees

Given a tree $A \leftarrow N' \rightarrow N \rightarrow A$, recall the explicit description of what a polynomial functor does:

$$\big(X_i \mid i \in A\big) \longmapsto \big(P_j \mid j \in A\big),$$

where

$$P_j = \sum_{n \in N_j} \prod_{e \in N'_n} X_{s(e)}$$

Note that if r is the root edge, then the variable X_r does not occur in the expression, since there is no node with input edge r. Also, if l is a leaf, then P_l is the constant empty set, simply because the fibre N_l is empty, and P_l is indexed over N_l. Suppose a is an edge which is not a leaf, then P_a is a sum indexed over all the dots whose output edge is a. There is precisely one such, say d (since we assumed a is not a leaf), so P_a is a monomial. it is the product, indexed over the fibre N'_d, which is the set of marked input edges of d, and for each such, the factor is the variable corresponding to that edge. So in short, P_a is the monomial

$$P_a = \prod_{e \text{ input of } a} X_e \qquad (12.2) \quad \boxed{\texttt{Pa}}$$

the product runs over child edges, i.e. edges coming in to the dot where a starts.

Note that each variable only occurs in at most one output monomial! since of course and edge is only the child edge of at most one other edge. As noticed already, the variable corresponding to the root edge does not occur at all.

In conclusion, the polynomial functor is a vector of monomials (12.2).

12.2.8 Example. Consider the dotless tree. It corresponds to the polynomial functor

$$\begin{tikzcd} & \emptyset \arrow[r] \arrow[dl] & \emptyset \arrow[dr] & \\ 1 & & & 1 \end{tikzcd}$$

which in turn is the constant polynomial functor $\emptyset$. (This is a one-variable polynomial functor, since there is only one edge in the tree.)

12.2.9 Example. Consider now the tree The polynomial functor is

$$\begin{tikzcd} & \emptyset \arrow[r] \arrow[dl] & 1 \arrow[dr] & \\ 1 & & & 1 \end{tikzcd}$$

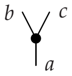
the constant polynomial 1. This is also a one-variable polynomial. Since there are no edges sitting over the single edge a, it is the empty monomial $X_a \mapsto X_a^{\emptyset} = 1$.

These two examples exhaust the one-variable trees. From the tree viewpoint, this is because they are the only trees with only one edge. From the viewpoint of the theorem characterising trees, it is because E is forced to be $\emptyset$ by condition (2), and B is forced to be 0 or 1 since t has to be injective.

12.2.10 Example. Consider the linear tree with n dots, and whose edges are decorated $0, 1, \ldots, n$. Then the corresponding polynomial functor P is the linear functor in $n+1$ variables, $P_i = X_{i+1}$ for $i \neq n$, and $P_n = 0$.

12.2.11 Example. Linear trees give linear polynomial functors, but of course most linear functors do not come from a tree.

12.2.12 Example. Consider now the little tree



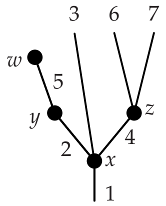
We have the polynomial functor in three variables X_a, X_b, X_c with components

$$P_a = X_b X_c, \qquad P_b = 0, \qquad P_c = 0.$$

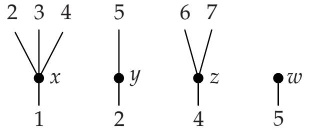
smalltree **12.2.13 Example.** It is an interesting exercise to take the polynomial functor associated to a tree and compose it with itself. Roughly this gives a polynomial functor whose set of operations is the full level-2 subtrees in the tree—together with the nullary nodes. That is, the only operations come from those nodes in the tree all of whose children are nodes again (not leaves). And then the nullary operations.

By this description, the result is a rather primitive polynomial functor for most small trees. . .

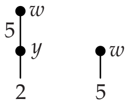
Example:



Then the operations of the composite polynomial functor is the set of all ways of decorating the four bouquets



with other bouquets. Note that *every* input edge of a bouquet needs a decoration. Hence it is impossible to decorate node x (the leaf 3 can't have a decoration), and also node z (since neither 6 nor 7 can be decorated). Node w can be decorated (by the empty decoration) and y can be decorated in a unique way, namely by w. (Slogan, *take those operations all of whose inputs can be filled with other operations*.) So in conclusion, the composite polynomial functor has two operations,



both nullary.

Of course the set of types is the same as for the original functor: $\{1,2,3,4,5,6,7\}$. We could compute the set $E$of partial operations by hand, but the fact that we keep careful track of arities and draw the operations as bouquets and trees with clear arity indication means that we can reconstruct the set E easily: since both operations are nullary, E is the empty set.

We could also compute the composite algebraically, thinking in terms of polynomial functions: we have

$$\begin{aligned}
P_1(\mathbf{X}) &= X_2X_3X_4\\
P_2(\mathbf{X}) &= X_5\\
P_3(\mathbf{X}) &= 0\\
P_4(\mathbf{X}) &= X_6X_7\\
P_5(\mathbf{X}) &= 1\\
P_6(\mathbf{X}) &= 0\\
P_7(\mathbf{X}) &= 0
\end{aligned}$$

so if we substitute P_i for X_i we get

$$\begin{aligned}
(P\circ P)_1(\mathbf{X}) &= 0\\
(P\circ P)_2(\mathbf{X}) &= 1\\
(P\circ P)_3(\mathbf{X}) &= 0\\
(P\circ P)_4(\mathbf{X}) &= 0\\
(P\circ P)_5(\mathbf{X}) &= 1\\
(P\circ P)_6(\mathbf{X}) &= 0\\
(P\circ P)_7(\mathbf{X}) &= 0
\end{aligned}$$

which agrees with the description obtained in graphical terms: it is constant since all operations are nullary, and the ones in the table correspond to the nullary operations.

12.3 The category *TEmb*

12.3.1 The category of trees and tree embeddings. Define the category ***TEmb*** to be the full subcategory of ***PolyEnd*** consisting of the trees. Hence a map of trees $\phi : \mathsf{S} \to \mathsf{T}$ is a diagram

$$\begin{tikzcd}
S^0 \arrow[d, "\phi^0"'] & S^2 \arrow[l] \arrow[r] \arrow[d, "\phi^2"'] \arrow[dr, phantom, "\lrcorner", very near start] & S^1 \arrow[r] \arrow[d, "\phi^1"] & S^0 \arrow[d, "\phi^0"] \\
T^0 & T^2 \arrow[l] \arrow[r] & T^1 \arrow[r] & T^0
\end{tikzcd}$$

(12.3) map

The cartesian condition amounts to 'arity preservation': the set of input edges of $b \in S^1$ maps bijectively onto the set of input edges of $\phi^1(b)$. Root and leaves are not in general preserved.

12.3.2 Lemma. *Morphisms in* ***TEmb*** *preserve the childhood relation. That is, for a morphism* $\phi : \mathsf{S} \to \mathsf{T}$*, if* x *is a child edge of* y *in* S *then* $\phi^0(x)$ *is a child edge of* $\phi^0(y)$ *in* T*. More generally, morphisms preserve distance.*

Proof. To say that x is a child of y means that x is not the root and $t(p(x)) = y$. The property of not being the root is preserved by any map (cf. commutativity of the left-hand square in the diagram), so $\phi^0(x)$ is not the root either. Now apply ϕ and use that it commutes with p and t, cf. (12.3). □

mono-cat **12.3.3 Proposition.** *Every morphism in* ***TEmb*** *is injective.*

Proof. Let $\phi : \mathsf{S} \to \mathsf{T}$ in ***TEmb***. Let $r \in T^0$ denote the image of the root edge. Let x, y be edges in S and suppose $\phi^0(x) = \phi^0(y)$. Since ϕ^0 preserves distance we have $d(x, 1) = d(\phi^0(x), r) = d(\phi^0(y), r) = d(y, 1)$. Since x and y have the same distance to the root, it makes sense to put $k := \min\{n \in \mathbb{N} \mid \sigma^n(x) = \sigma^n(y)\}$, and $z := \sigma^k(x) = \sigma^k(y)$ (nearest common ancestor). If $k > 0$, then the edges $\sigma^{k-1}(x)$ and $\sigma^{k-1}(y)$ are both children of z, and by childhood preservation, we have $\phi(\sigma^{k-1}(x)) = \phi(\sigma^{k-1}(y))$. But ϕ induces a bijection between the fibre $(S^2)_z$ and the fibre $(T^2)_{\phi^0(z)}$, so we conclude that already $\sigma^{k-1}(x) = \sigma^{k-1}(y)$, contradicting the minimality of k. Hence $k = 0$, which is to say that already $x = y$. Hence we have shown that ϕ^0

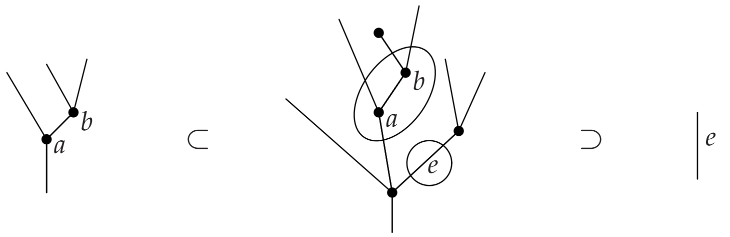
is injective. Since t is always injective, it follows that also ϕ^1 and ϕ^2 are injective. □

The proposition shows that the category ***TEmb*** is largely concerned with the combinatorics of subtrees, which we now pursue. It must be noted, though, that the category contains nontrivial automorphisms. In particular it is easy to see that

12.3.4 Lemma. *The assignment of a one-node tree to every finite set as in 12.2.5 defines a fully faithful functor from the groupoid of finite sets and bijections into **TEmb**. (The essential image consists of the trees with precisely one node.)* □

12.3.5 Subtrees. A subtree of a tree T is an isomorphism class of arrows $\mathsf{S} \to \mathsf{T}$ in ***TEmb***; more concretely it is an arrow $\mathsf{S} \to \mathsf{T}$ for which each of the three set maps are subset inclusions. Translating into classical viewpoints on trees, subtree means connected subgraph with the property that if a node is in the subgraph then all its incident edges are in the subgraph too.

Here are two examples:



edges

12.3.6 Edges. For each edge x of T there is a subtree ı $\to \mathsf{T}$ given by

$$\begin{tikzcd}
1 \arrow[d, "\ulcorner x \urcorner"'] & 0 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & 0 \arrow[r] \arrow[d] & 1 \arrow[d, "\ulcorner x \urcorner"] \\
T^0 & T^2 \arrow[l] \arrow[r] & T^1 \arrow[r] & T^0 .
\end{tikzcd}$$

The subtree consists solely of the edge x. The edge is the root edge iff the left-hand square is a pullback, and the edge is a leaf iff the right-hand square is a pullback.

12.3.7 One-node subtrees. For each node b in T there is a subtree inclusion

$$\begin{tikzcd}
(T^2)_b + 1 \arrow[d] & (T^2)_b \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \{b\} \arrow[r] \arrow[d] & (T^2)_b + 1 \arrow[d] \\
T^0 & T^2 \arrow[l] \arrow[r] & T^1 \arrow[r] & T^0
\end{tikzcd}$$

The vertical maps at the ends are the sum of $s \mid (T^2)_b$ and the map sending 1 to $t(b)$. The subtree defined is the local one-node subtree at b: the node itself with all its incident edges.

determinedbynodes

12.3.8 Proposition. *Let R and S be nontrivial subtrees in T, and suppose that $R^1 \subset S^1$. Then $\mathsf{R} \subset \mathsf{S}$. In particular, a nontrivial subtree is determined by its nodes.*

Proof. We need to provide the dotted arrows in the diagram

$$\begin{tikzcd}
R^0 \arrow[dd] \arrow[dr, dashed] & & R^2 \arrow[ll] \arrow[rr] \arrow[dd] \arrow[dr, dashed] & & R^1 \arrow[rr] \arrow[dd] \arrow[dr, dashed] & & R^0 \arrow[dd] \arrow[dr, dashed] & \\
& S^0 \arrow[dl] & & S^2 \arrow[ll] \arrow[rr] \arrow[dl] & & S^1 \arrow[rr] \arrow[dl] & & S^0 \arrow[dl] \\
T^0 & & T^2 \arrow[ll] \arrow[rr] & & T^1 \arrow[rr] & & T^0 &
\end{tikzcd}$$

The arrow $R^1 \to S^1$ is the assumed inclusion of nodes. For each node b in R we have a bijection between the fibre $(R^2)_b$ and the fibre $(S^2)_b$. These bijections assemble into a map $R^2 \to S^2$ and a cartesian square. Since $R^0 = R^2 + \{r\}$ where r is the root edge of R, to specify the arrow $R^0 \to S^0$ it remains to observe that r maps into S^0: indeed, there is a $b \in R^1$ with $t(b) = r$. Hence $\phi^0(r) = \phi^0(t(b)) = t(\phi^1(b)) \in S^0$. □

ideal

12.3.9 Ideal subtrees. An *ideal subtree* is a subtree containing all the descendant nodes of its edges, and hence also all the descendant edges. (Hence it is a 'down-set' for the tree order (both with respect to nodes and with respect to edges), and just by being a subtree it is also closed under binary join.)

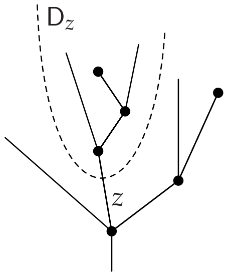
Each edge z of a tree T determines an ideal subtree denoted D_z:

$$\begin{tikzcd}
\mathsf{D}_z : & D^0 \arrow[d, hook] & D^2 \arrow[l] \arrow[r] \arrow[d, hook] \arrow[dr, phantom, "\lrcorner", very near start] & D^1 \arrow[r] \arrow[d, hook] \arrow[dr, phantom, "\lrcorner", very near start] & D^0 \arrow[d, hook] \\
& T^0 & T^2 \arrow[l] \arrow[r] & T^1 \arrow[r] & T^0
\end{tikzcd}$$

where

$$\begin{aligned}
D^0 &:= \{x \in T^0 \mid x \leq z\}, \\
D^1 &:= \{b \in T^1 \mid t(b) \in D^0\}, \\
D^2 &:= \{e \in T^2 \mid t(p(e)) \in D^0\} = D^0 \smallsetminus \{z\}.
\end{aligned}$$

It is easy to check that this is a tree; it looks like this:



Note also that we have $x \in \mathsf{D}_z \Leftrightarrow x \leq z$.

ideal3 **12.3.10 Lemma.** *The following are equivalent for a tree embedding $\phi : \mathsf{S} \to \mathsf{T}$:*

1. *The image subtree is an ideal subtree.*
2. *The right-hand square is cartesian (like in the above diagram).*
3. *The image of each leaf is again a leaf.*

Proof. (1) ⇒ (2): clearly every ideal subtree $\mathsf{S} \subset \mathsf{T}$ is equal to D_z for z the root of S. Hence the embedding has cartesian right-hand square.

(2) ⇒ (3): a leaf in S is characterised (12.3.6) as an edge for which the right-hand square is cartesian; composing with ϕ gives then again a cartesian right-hand square, so the edge is again a leaf in T.

(3) ⇒ (1): let x be an edge in S, having a child node b in T (that is, $p(b) = x$). This means x is not a leaf in T, and hence by assumption, not a leaf in S either. So b is also in S. □

12.3.11 Pruning. Using complements, it is not difficult to see that an edge $z \in T^0$ defines also another subtree which has the original root, but where all descendants of z have been pruned. In other words, the ideal subtree D_z is thrown away (except for the edge z itself). Formally, with the notation of the ideal subtree: put $C^1 := T^1 \smallsetminus D^1$ and $C^2 := T^2 \smallsetminus D^2$. Then clearly we have a cartesian square

$$\begin{tikzcd} C^2 \arrow[r] \arrow[d, hook] \arrow[dr, phantom, "\lrcorner", very near start] & C^1 \arrow[d, hook] \\ T^2 \arrow[r] & T^1. \end{tikzcd}$$

Now simply put $C^0 := C^2 + \{1\}$ (the original root). It remains to see that the map $t : T^1 \to T^0$ restricts to $C^1 \to C^0$, but this follows from the fact that if $t(b)$ is not in D^0, then it must be in either C^2 or 1. Using simple set theory, one readily checks that this is a tree again.

In any poset, we say that two elements e and e' are *incomparable* if neither $e \leq e'$ nor $e' \leq e$. If two subtrees have incomparable roots then they are disjoint. Indeed, suppose the subtrees S and S' of T have an edge x in common. Then the totally ordered set of ancestors of x in T will contain both the root of S and the root of S', hence they are comparable. Clearly siblings are incomparable. In particular, if two subtrees have sibling roots, then they are disjoint.

incomparable **12.3.12 Lemma.** *Let x and y be edges of a tree T. Then the following are equivalent:*

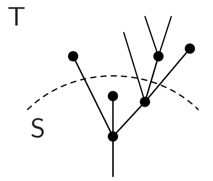
1. *The ideal subtrees D_x and D_y are disjoint.*

2. *x and y are incomparable (i.e. neither $x \leq y$ nor $y \leq x$).*

3. *There exists a subtree in which x and y are leaves.*

Proof. If $x \leq y$ then clearly $\mathsf{D}_x \subset \mathsf{D}_y$. On the other hand if D_x and D_y have an edge e in common, then $e \leq x$ and $e \leq y$, and hence $x \leq y$ or $y \leq x$. Concerning condition (3): if x and y are leaves of a subtree, in particular they are both minimal, and in particular they are incomparable. Conversely, if they are incomparable, then we already know that the ideal subtrees they generate are disjoint, so we can prune at x and y to get a subtree in which x and y are leaves. □

12.3.13 Root-preserving embeddings. An arrow $\mathsf{S} \to \mathsf{T}$ in ***TEmb*** is called *root preserving* if the root is mapped to the root. In other words, S viewed as a subtree of T contains the root edge of T:



The root preserving subtrees are those that are up-sets in the tree order. It is easy to check that $\mathsf{S} \to \mathsf{T}$ is root-preserving if and only if the left-hand square is a pullback.

ideal-root **12.3.14 Lemma.** *If a tree embedding is both root preserving and ideal, then it is invertible (i.e. its image is the whole tree).*

Proof. Indeed, if it is root preserving then its image contains 1, and because it is ideal its image contains all other edges, as they are descendants of the root. □

root-ideal **12.3.15 Proposition.** *Every arrow $\phi : \mathsf{S} \to \mathsf{T}$ in* ***TEmb*** *factors uniquely as a root-preserving map followed by an ideal embedding.*

Proof. Put $r := \phi^0(\text{root})$, and consider the ideal subtree $\mathsf{D}_r \subset \mathsf{T}$. Since the map preserves childhood relation, it is clear that all edges in S map into D_r, and this map is root preserving by construction. □

12.3.16 Remark. One can equally well factor every map the other way around: first an ideal embedding and then a root-preserving embedding. We will not have any use of that factorisation, though.

frame **12.3.17 Lemma.** *A subtree is determined by its boundary.*

Proof. Let $\mathsf{S} \subset \mathsf{T}$ and $\mathsf{S}' \subset \mathsf{T}$ be subtrees with common boundary. Suppose b is a node of S which is not in S'. Since b is in S, for some k we have $\sigma^k(t(b)) = \text{root}(\mathsf{S}) = \text{root}(\mathsf{S}')$. In this chain of nodes and edges, there is a node b which is in S but not in S', and such that $t(b)$ is an edge in S'. This means $t(b)$ is a leaf in S' and hence a leaf in S, but this in turn implies that b is not in S, in contradiction with the initial assumption. So the two subtrees contain the same nodes. If they do contain nodes at all then they are equal by Lemma 12.3.8. If both subtrees are trivial, then they must coincide because their roots coincide. □

12.3.18 Pushouts in *PolyEnd*. A polynomial functor P is a diagram in ***Set*** of shape

$$\cdot \leftarrow \cdot \rightarrow \cdot \rightarrow \cdot$$

While pointwise sums are also sums in ***PolyEnd***, pointwise pushouts are not in general pushouts in ***PolyEnd***, because of the condition on arrows that the middle square be cartesian. Only pushouts over polynomial functors of shape $? \leftarrow 0 \rightarrow 0 \rightarrow ?$ can be computed pointwise. In particular we can take pushouts over the trivial tree $\shortmid : 1 \leftarrow 0 \rightarrow 0 \rightarrow 1$. The pushout of the morphisms $\mathsf{S} \leftarrow \ \shortmid \ \rightarrow \mathsf{T}$ is the polynomial endofunctor given by

$$\begin{tikzcd}
 & & S^2 + T^2 \arrow[r] \arrow[dl] & S^1 + T^1 \arrow[dr] & & \\
 & S^0 + T^0 \arrow[dl] & & & S^0 + T^0 \arrow[dr] & \\
S^0 +_1 T^0 & & & & & S^0 +_1 T^0,
\end{tikzcd} \tag{12.4}$$

pushout-diagra

where $S^0 +_1 T^0$ denotes the amalgamated sum over the singleton.

pushout

12.3.19 Proposition. *Given a diagram of trees and tree embeddings*

$$\begin{tikzcd}
\mathsf{S} & \shortmid \arrow[l, "r"'] \arrow[r, "l"] & \mathsf{T}
\end{tikzcd}$$

*such that r is the root edge in S, and l is a leaf in T, then the pushout in **PolyEnd** is again a tree, called the grafting of S onto the leaf of T, and denoted $\mathsf{S} +_{\shortmid} \mathsf{T}$.*

Proof. We check that the polynomial endofunctor (12.4) is a tree by inspection of the four axioms. Axiom 1: it is obvious the involved sets are finite. Axiom 2: we check that the right-hand leg is injective: to say that l is a leaf of T means it is not in the image of $t : T^1 \rightarrow T^0$. So we can write $S^1 + T^1 = S^1 +_{\{l\}} (\{l\} + T^1)$, and the map we want to prove injective is just the inclusion $S^1 + T^1 = S^1 +_{\{l\}} (\{l\} + T^1) \hookrightarrow S^1 +_{\{l\}} T^0$. Axiom

3: we check that the left-hand leg is injective and has singleton complement: this follows from the calculation $S^0 +_1 T^0 = (S^2 + \{r\}) +_{\{r\}} T^0 = S^2 + T^0 = S^2 + T^2 + 1$ (where 1 denotes the root of the bottom tree T) Axiom 4: we check the walk-to-the-root condition: for $x \in S^0$, in a finite number of steps we arrive at $r = e = l$, and from here in another finite number of steps we come down to the root of T. □

12.3.20 Remark. More generally, the pushout of a root-preserving embedding along an ideal embedding is again a tree, and the two resulting maps are again root-preserving and ideal, respectively, as in this diagram

$$\begin{tikzcd}
\cdot \arrow[r, "\text{root pres.}"] \arrow[d, "\text{ideal}"'] & \cdot \arrow[d, dashed, "\text{ideal}"] \\
\cdot \arrow[r, dashed, "\text{root pres.}"'] & \cdot \arrow[ul, phantom, "\ulcorner", very near start]
\end{tikzcd}$$

We will not need or prove this result here.

The following expresses the recursive characterisation of trees.

recursive

12.3.21 Proposition. *A tree* T *is either a nodeless tree, or it has a node $b \in T^1$ with $t(b) = 1$; in this case for each $e \in (T^2)_b$ consider the ideal subtree* D_e *corresponding to e. Then the original tree* T *is the grafting of all the* D_e *onto the input edges of b.*

Proof. The grafting exists by Proposition 12.3.19, and is a subtree in T by the universal property of the pushout. Clearly every node in T is either b or a node in one of the ideal subtrees, therefore the grafting is the whole tree, by Lemma 12.3.8. □

12.3.22 Corollary. *An automorphism of a tree amounts to permutation of siblings whose generated ideal subtrees are isomorphic.*

Proof. Use the recursive characterisation of trees. By childhood preservation, an automorphism must send an edge e to a sibling e'. For the same reason it must map D_e isomorphically onto $\mathsf{D}_{e'}$. □

12.3.23 Inner edges. An *inner edge* of a tree

$$\begin{tikzcd}
T^0 & T^2 \arrow[l, "s"'] \arrow[r, "p"] & T^1 \arrow[r, "t"] & T^0
\end{tikzcd}$$

is one that is simultaneously in the image of s and t. In other words, the set of inner edges is naturally identified with $T^1 \times_{T^0} T^2$ considered as a subset of T^0; its elements are pairs (b, e) such that $t(b) = s(e)$.

graft-onenode

12.3.24 Corollary. *Every nontrivial tree* T *is the grafting (indexed by the set of inner edges* $T^1 \times_{T^0} T^2$*) of its one-node subtrees.* □

The *elements* of a tree T are its nodes and edges. i.e. its elementary subtrees. These form a poset ordered by inclusion, and we denote this category $\mathrm{el}(\mathsf{T})$. There is an obvious functor $\mathrm{el}(\mathsf{T}) \to$ ***TEmb***. This functor has a colimit which is just T. Indeed, each edge is included in at most two one-node subtrees of T, and always as root in one and as leaf in the other; the colimit is obtained from these pushouts. The general notion of elements of a polynomial endofunctor will be introduced in Section **??**.

12.4 P-trees

The trees studied so far are in a precise sense abstract trees, whereas many trees found in the literature are structured trees, amounting to a morphism to a fixed polynomial functor. The structure most commonly found is planarity: a planar structure on a tree T is a linear ordering of the input edges of each node, i.e. a linear ordering on each fibre of $T^2 \to T^1$. This amounts to giving a morphism $\mathsf{T} \to \mathsf{M}$, where M is the free-monoid monad (1.2.8).

P-trees

12.4.1 P-trees. Let P be a fixed polynomial endofunctor $P^0 \leftarrow P^2 \to P^1 \to P^0$. By definition, the category of P-*trees* is the comma category ***TEmb*** $/\mathsf{P}$ whose objects are trees T with a morphism $\mathsf{T} \to \mathsf{P}$ in ***PolyEnd***. Explicitly, a P-tree is a tree $T^0 \leftarrow T^2 \to T^1 \to T^0$ together with a diagram

$$\begin{tikzcd}
T^0 \arrow[d] & T^2 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & T^1 \arrow[r] \arrow[d] & T^0 \arrow[d] \\
P^0 & P^2 \arrow[l] \arrow[r] & P^1 \arrow[r] & P^0 .
\end{tikzcd}$$

Unfolding further the definition, we see that a P-tree is a tree whose edges are decorated in P^0, whose nodes are decorated in P^1, and with the additional structure of a bijection for each node $n \in T^1$ (with decoration $b \in P^1$) between the set of input edges of n and the fibre $(P^2)_b$, subject to the compatibility condition that such an edge $e \in (P^2)_b$ has decoration $s(e)$, and the output edge of n has decoration $t(b)$. Note that the P^0-decoration of the edges is completely specified by the node decoration together with the compatibility requirement, except for the case of a nodeless tree. (The notion of P-tree for a polynomial endofunctor P is closely related to the notion of T_S-tree of Baez and Dolan [11, Proof of Thm. 14], but they neglect to decorate the edge in the nodeless tree.)

If P is the identity monad, a P-tree is just a linear tree. If P is the free-monoid monad, a P-tree is precisely a planar tree, as mentioned. If P is the free-nonsymmetric-operad monad on $\boldsymbol{Set}/\mathbb{N}$, the P-trees are the 3-dimensional opetopes, and so on: opetopes in arbitrary dimension are P-trees for a suitable P, cf. [11], [75, §7.1], [64].

12.4.2 Remark. It is important to note that P-trees are something genuinely different from just trees, in the sense that ***TEmb*** is not equivalent to ***TEmb***/P for any P. It is true of course that every tree admits a planar structure, i.e. a decoration by the free-monoid monad M (1.2.8): the possible diagrams

$$\begin{tikzcd}
T^0 \arrow[d] & T^2 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & T^1 \arrow[r] \arrow[d] & T^0 \arrow[d] \\
I & \mathbb{N}' \arrow[l] \arrow[r] & \mathbb{N} \arrow[r] & 1
\end{tikzcd}$$

have to send a node $b \in T^1$ to its arity n (the number of input edges), and then there are $n!$ different choices for mapping the fibre to the n-element set $\mathbf{n}$, the fibre over n.

The crucial property of P-trees is that they are rigid:

12.4.3 Proposition. *P-trees have no nontrivial automorphisms.*

Proof. Every automorphism of a tree consists in permuting siblings. Now in a P-tree, the set of siblings (some set $(T^2)_b$) is in specified bijection with $(P^2)_{\phi^1(b)}$, so no permutations are possible. □

12.4.4 Proposition. *P-trees have no automorphisms.*

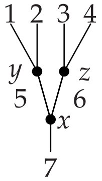
Basically the reason is that the input edges of a node can not be permuted because they are in fixed bijection with some fibre E_b of the decorating endofunctor P.

We should check the details. One might think perhaps that this statement holds more generally for any slice category $\boldsymbol{Poly}/P$. But this is not true: if P is the polynomial functor with only one operation, and this operation is of arity 2 (so it is the one-variable polynomial functor represented by the map $E := \{\texttt{left}, \texttt{right}\} \to \{\texttt{continue}\}$), then we can take another polynomial functor over it, $E \times 2 \to 2$. This polynomial functor has an involution over P, namely the involution of 2. . .

Even simpler: over the identity functor $1 \leftarrow 1 \to 1 \to 1$, the polynomial functor $1 \leftarrow 7 \to 7 \to 1$ has 7! auts!

This can not happen for trees: an isomorphism of trees can only permute siblings, and this is not possible when the siblings are in bijection with some fixed fibre E_b. More formally, it is not difficult to see that a isomorphism diagram must be compatible with the walk-to-the-root functions, and hence it can only permute siblings. . .

Here is an example:



With the bijections to $\{\texttt{left}, \texttt{right}\}$ as indicated by the orientation of the paper. Let's try to interchange y and z, and their subtrees, so we also interchange $1 \leftrightarrow 3$ and $2 \leftrightarrow 4$. But since we interchange y and z we are forced also to interchange their images under the out-edge map, namely 5 and 6. But this is impossible because they constitute the fibre over x, and hence they must be in fixed bijection with $\{\texttt{left}, \texttt{right}\}$. . .

The basic results about trees, notably grafting (12.3.19) and the recursive characterisation (12.3.21), have obvious analogues for P-trees. We shall not repeat those results.

12.4.5 The set of P-trees. Let P be a polynomial endofunctor. Denote by $\mathrm{tr}(\mathsf{P})$ the set of isomorphism classes of P-trees, i.e. isomorphism classes of diagrams

$$\begin{tikzcd}
T^0 \arrow[d] & T^2 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & T^1 \arrow[r] \arrow[d] & T^0 \arrow[d] \\
P^0 & P^2 \arrow[l] \arrow[r] & P^1 \arrow[r] & P^0
\end{tikzcd}$$

where the first row is a tree. Note that $\mathrm{tr}(\mathsf{P})$ is naturally a set over P^0 by returning the decoration of the root edge.

fix **12.4.6 Theorem.** *If* P *is a polynomial endofunctor then* $\mathrm{tr}(\mathsf{P})$ *is a least fixpoint (i.e. initial Lambek algebra) for the endofunctor*

$$\begin{aligned} 1 + \mathsf{P} : \textbf{\textit{Set}}/P^0 &\longrightarrow \textbf{\textit{Set}}/P^0 \\ X &\longmapsto P^0 + \mathsf{P}(X). \end{aligned}$$

Proof. The proof uses the recursive characterisation of P-trees analogous to 12.3.21. For short, put $W := \mathrm{tr}(\mathsf{P})$. We have

$$\mathsf{P}(W) = \left\{ (b,f) \mid b \in P^1, \begin{tikzcd}(P^2)_b \arrow[rr, "f"] \arrow[dr] & & W \arrow[dl] \\ & P^0 & \end{tikzcd} \right\}$$

This set is in natural bijection with the set of P-trees with a root node decorated by $b \in P^1$. Indeed, given $(b,f) \in \mathsf{P}(W)$, we first consider the unique one-node P-tree whose node is decorated by b. (This is well-defined: since $(P^2)_b$ is finite, the one-node tree is given as in 12.2.5, and the decorations are completely determined by the requirement that the node is decorated by b.) Now for each $e \in (P^2)_b$ we can graft the P-tree $f(e)$ onto the leaf e of that one-node P-tree as in 12.3.19. The result is a P-tree D with root node decorated by b. Conversely, given a P-tree D with root node decorated by b, define $f : (P^2)_b \to W$ by sending e to the ideal sub-P-tree D_e.

Now, W is the sum of two sets: the nodeless P-trees (these are in bijection with P^0) and the P-trees with a root node. Hence we have $(1 + \mathsf{P})(W) \xrightarrow{\sim} W$, saying that W is a fixpoint.

Finally, we must show that W is a *least* fixpoint. Suppose $V \subset W$ is also a fixpoint. Let $W_n \subset W$ denote the set of P-trees with at most n nodes. Clearly $W_0 \subset V$. But if $W_n \subset V$ then also $W_{n+1} \subset V$ because each tree with $n+1$ nodes arises from some (b,f) where b decorates the root node and $f : (P^2)_b \to W_n$. □

12.4.7 Historical remarks: well-founded trees. Theorem 12.4.6 has a long history: it is a classical observation (due to Lambek) that the elements of an initial algebra for an endofunctor P are tree-like structures, and that the branching profile of such trees depends on P. A very general version of the theorem is due to Moerdijk and Palmgren [81] providing categorical semantics for the notion of W-types (wellfounded trees) in Martin-Löf type theory. Briefly, under the Seely correspondence between (extensional) type theory and locally cartesian closed categories $\mathscr{E}$, the Sigma and Pi types correspond to dependent sums and products (as in (8.1)). The W type constructor associates to a given combination P of Sigma and Pi types a new inductive type W_{P}. Under the correspondence, P is a polynomial endofunctor on $\mathscr{E}$ (i.e. with P^0 terminal), and W_{P} is its initial algebra.

The new feature of Theorem 12.4.6 (and the treatment leading to it) is to have trees and endofunctors on a common footing. This makes everything more transparent. Such a common footing was not possible in [81] because they only considered polynomial endofunctors P with P^0 terminal. Trees cannot be captured by such, since it is essential to be able to distinguish the edges in a tree. The case of arbitrary polynomial functors was considered by Gambino and Hyland [38], corresponding to dependent type theory.

We shall come back to trees, and notably show that the free monad on a tree $A \leftarrow M \rightarrow N \rightarrow A$ is given by

$$P^0 \leftarrow \mathsf{tr}'(\mathsf{P}) \rightarrow \mathsf{tr}(\mathsf{P}) \rightarrow P^0$$

ch

Chapter 13

Polynomial monads

Ch:II:monads

This chapter has been superseded by Gambino–Kock [39]: In fact you should rather read that paper.

13.1 The free polynomial monad on a polynomial endofunctor

CONDENSE THIS! SINCE MOST OF THE ARGUMENTS WERE ALREADY GIVEN IN THE ONE-VARIABLE CASE

13.1.1 Polynomial monads. A *polynomial monad* is a polynomial endofunctor $P : \mathbf{Set}/I \to \mathbf{Set}/I$ equipped with a composition law $\mu : P \circ P \to P$ with unit $\eta : \mathsf{Id} \to P$, satisfying the usual associativity and unit conditions. We require the structure maps μ and η to be cartesian natural transformations (so we should perhaps rather call this notion cartesian polynomial monad).

13.1.2 Graphical interpretation. The composition law is described graphically as an operation of contracting trees (formal compositions of bouquets) to bouquets. If P is given by $I \leftarrow E \rightarrow B \rightarrow I$, we shall refer to I as the set of *types* of P, and B as the set of *operations*. Since we have a unit, we can furthermore think of E as the set of *partial operations*, i.e. operations all of whose inputs except one are fed with a unit. The composition law can be described in terms of partial operations as a map

$$B \times_I E \to B,$$

consisting in substituting one operation into one input of another operation, provided the types match: $t(b) = s(e)$.

Fix a set I, and consider polynomial monads on $\boldsymbol{Set}/I$. These are just monoids in the monoidal category $\boldsymbol{PolyFun}^{\mathrm{c}}(I)$. Let $P : \boldsymbol{Set}/I \to \boldsymbol{Set}/I$ denote a polynomial functor represented by

$$I \leftarrow E \to B \to I.$$

13.1.3 The free monad on a polynomial endofunctor. Given a polynomial endofunctor $P : \boldsymbol{Set}/I \to \boldsymbol{Set}/I$, a *$P$-algebra* is a pair (X, a) where X is an object of $\boldsymbol{Set}/I$ and $a : P(X) \to X$ is an arrow in $\boldsymbol{Set}/I$ (not subject to any further conditions). A *P-algebra map* from (X, a) to (Y, b) is an arrow $f : X \to Y$ giving a commutative diagram

$$\begin{tikzcd}
P(X) \arrow[r, "P(f)"] \arrow[d, "a"'] & P(Y) \arrow[d, "b"] \\
X \arrow[r, "f"'] & Y.
\end{tikzcd}$$

Let P-$\boldsymbol{alg}/I$ denote the category of P-algebras and P-algebra maps. The forgetful functor $U : P\text{-}\boldsymbol{alg}/I \to \boldsymbol{Set}/I$ has a left adjoint F, the *free P-algebra functor*. The monad $T := U \circ F : \boldsymbol{Set}/I \to \boldsymbol{Set}/I$ is the *free monad* on P. This is a polynomial monad, and its set of operations is the set of P-trees, as we now explain.

Recall that a P-tree is a tree with edge set A, node set N, and node-with-marked-input-edge set N', together with a diagram

$$\begin{tikzcd}
A \arrow[d, "\alpha"'] & N' \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & N \arrow[r] \arrow[d, "\beta"] & A \arrow[d, "\alpha"] \\
I & E \arrow[l] \arrow[r] & B \arrow[r] & I
\end{tikzcd}$$

The P-trees are obtained by freely grafting elements of B onto the leaves of elements of B, provided the decorations match (and formally adding a dotless tree for each $i \in I$). More formally, the set of isomorphism classes of P-trees, which we denote by $\mathrm{tr}(P)$, is a least fixpoint for the polynomial endofunctor

$$\begin{array}{ccc}
\boldsymbol{Set}/I & \longrightarrow & \boldsymbol{Set}/I \\
X & \longmapsto & I + P(X);
\end{array}$$

it is given explicitly as the colimit

$$\mathrm{tr}(P) = \bigcup_{n\in\mathbb{N}} (I+P)^n(\emptyset).$$

13.1.4 Explicit description of the free monad on P. A slightly more general fixpoint construction characterises the free P-algebra monad T: if A is an object of ***Set***$/I$, then $T(A)$ is a least fixpoint for the endofunctor $X \mapsto A + P(X)$. In explicit terms,

$$T(A) = \bigcup_{n\in\mathbb{N}} (A+P)^n(\emptyset).$$

It is the set of P-trees with input edges decorated in A. But this is exactly the characterisation of a polynomial functor with operation set $\mathrm{tr}(P)$: let $\mathrm{tr}'(P)$ denote the set isomorphism classes of P-trees with a marked input leaf, then $T :$ ***Set***$/I \to$ ***Set***$/I$ is the polynomial functor given by

$$\begin{tikzcd}
 & \mathrm{tr}'(P) \arrow[r] \arrow[dl] & \mathrm{tr}(P) \arrow[dr] & \\
I & & & I\,.
\end{tikzcd}$$

The maps are the obvious ones: return the marked leaf, forget the mark, and return the root edge, respectively. The monad structure of T is described explicitly in terms of grafting of trees. In a partial-composition description, the composition law is

$$\mathrm{tr}(P) \times_I \mathrm{tr}'(P) \to \mathrm{tr}(P)$$

consisting in grafting a tree onto the specified input leaf of another tree. The unit is given by $I \to \mathrm{tr}(P)$ associating to $i \in I$ the dotless tree with edge decorated by i. (One can readily check that this monad is cartesian.)

The above should amount to the following theorem:

freeI **13.1.5 Theorem.** *(cf. [38], [39].) The forgetful functor* **PolyMnd**$(I) \to$ **PolyFun**$^c(I)$ *has a left adjoint, denoted* $\mathsf{P} \mapsto \overline{\mathsf{P}}$. *The monad* $\overline{\mathsf{P}}$ *is the* free monad *on* P.

An explicit construction of $\overline{\mathsf{P}}$ is given in 14.1.1.

This involves checking this:

13.1.6 Proposition. *The free monad on a polynomial functor is cartesian.*

Proof. Check if this is not completely general... Otherwise, the proof of the one-variable case carries over almost verbatim. □

INCLUDE THE PROOF OF CARTESIANNESS HERE

Suppose $\mathcal{C}$ is a locally presentable category and F is an ω-accessible endofunctor of $\mathcal{C}$. Then the free monad on F exists, and its underlying endofunctor has the description

$$\overline{\mathsf{F}} = \operatorname*{colim}_n F_n$$

where $F_0 := \mathrm{id}$ and $F_{n+1} := \mathrm{id} + (F \circ F_n)$.

Remark: this description is not the most obvious one, which would start instead with $F_0 := \emptyset$ (and then same recursive description). The following arguments work for both definitions, but $F_0 = \mathrm{id}$ actually has better properties.

Lemma. Suppose furthermore that F and G are cartesian. If $u : F \Rightarrow G$ is a cartesian natural transformation, then the induced natural transformation $\overline{\mathsf{u}} : \overline{\mathsf{F}} \Rightarrow \overline{\mathsf{G}}$ is again cartesian.

Proof. The natural transformation $\overline{\mathsf{u}}$ is the colimit of the sequence of natural transformations $u_n : F_n \Rightarrow G_n$ with $u_0 := \mathrm{id}_{\mathrm{id}}$ and $u_{n+1} : F_{n+1} \to G_{n+1}$ defined as the composite

$$\mathrm{id} + (F \circ F_n) \overset{\mathrm{id} + F(u_n)}{\Rightarrow} \mathrm{id} + (F \circ G_n) \overset{\mathrm{id} + u}{\Rightarrow} \mathrm{id} + (G \circ G_n).$$

Each u_n is a cartesian natural transformation. Indeed, $u_0 = \mathrm{id}$ clearly is, and if u_n is then so is u_{n+1} since F preserves pullbacks and u is cartesian. To see that the colimit $\overline{\mathsf{u}}$ is a cartesian natural transformation, we must check that for a given arrow $x \to y$ the naturality square

$$\begin{tikzcd}
\overline{\mathsf{F}}x \arrow[r] \arrow[d] & \overline{\mathsf{F}}y \arrow[d] \\
\overline{\mathsf{G}}x \arrow[r] & \overline{\mathsf{G}}y
\end{tikzcd}$$

is a pullback. We compute

$$\begin{aligned}\overline{\mathsf{G}}x \times_{\overline{\mathsf{G}}y} \overline{\mathsf{F}}y &\simeq (\operatorname*{colim}_n G_n x) \times_{\operatorname{colim}_n G_n y} (\operatorname*{colim}_n F_n y)\\ &\simeq \operatorname*{colim}_n (G_n x \times_{G_n y} F_n y)\\ &\simeq \operatorname*{colim}_n F_n x\\ &\simeq \overline{\mathsf{F}}x.\end{aligned}$$

Here we used the description of the free monad, the fact that finite limits distribute over filtered colimits in locally presentable categories, and the fact that each $u_n : F_n \Rightarrow G_n$ is cartesian.

I wanted to understand the cartesian condition also in the partial substitution viewpoint...

13.1.7 Alternative fixpoint construction. There is an alternative description of the free monad, which we briefly mention: I HAVE NOT YET FIGURED OUT THE RELATIONSHIP BETWEEN THE TWO FIXPOINT CONSTRUCTIONS. The free monad is a least fixpoint for the functor

$$\begin{aligned}\Gamma := \Gamma_P : \boldsymbol{Poly}^{\mathrm{c}}(I) &\longrightarrow \boldsymbol{Poly}^{\mathrm{c}}(I)\\ Q &\longmapsto \mathsf{Id} \,+\, P \circ Q.\end{aligned}$$

Let $\emptyset$ denote the constant polynomial functor on the empty set (it is given by $\emptyset \to \emptyset$). One can check that Γ preserves monos, hence there is a sequence of monos

$$\emptyset \hookrightarrow \Gamma(\emptyset) \hookrightarrow \Gamma^2(\emptyset) \hookrightarrow \ldots$$

If P is finitary (i.e., $p : E \to B$ is a finite map—this will always be the case below) then it preserves such sequential colimits, and the least fixpoint can be constructed as

$$\overline{\mathsf{P}} = \bigcup_{n \geq 0} \Gamma^n(\emptyset).$$

The equation satisfied by $\overline{\mathsf{P}}$,

$$\overline{\mathsf{P}} = \mathsf{Id} \,+\, P \circ \overline{\mathsf{P}},$$

expresses the recursive characterisation of P-trees: a P-tree is either a dotless tree (decorated by an element in I) or a finite set of P-trees.

Compare this construction with the one in Section 4.3: there we constructed an initial algebra for a polynomial endofunctor, i.e. a least fixpoint for this endofunctor. The result was basically just a set, and typically a set of dead trees of some sort.

Now we are constructing a new endofunctor, as a fixpoint in $\boldsymbol{Poly}^{\mathrm{c}}$. This endofunctor then has a set of operations, and the elements in this set are trees, but trees with non-zero arities, live trees!

13.2 Monads in the double category setting

We know what polynomial monads are: they are polynomial endofunctors $M : I \leftarrow E \rightarrow B \rightarrow I$ equipped with cartesian natural transformations $\mathsf{Id}_I \Rightarrow M \Leftarrow M \circ M$. So for each I we have a category of polynomial monads $\boldsymbol{PolyMnd}(I)$. The arrows in this category are monad maps: these are cartesian natural transformations $M_0 \Rightarrow M_1$ such that this diagram commutes:

$$\begin{tikzcd}
I \arrow[d, equal] \arrow[r, Rightarrow] & M_0 \arrow[d, Rightarrow] & M_0 \circ M_0 \arrow[l, Rightarrow] \arrow[d, Rightarrow] \\
I \arrow[r, Rightarrow] & M_1 & M_1 \circ M_1 \arrow[l, Rightarrow]
\end{tikzcd}$$

Claim. This can also be expressed in a partial-composition-law fashion:

$$\begin{tikzcd}
B_0 \times_I E_0 \arrow[r] \arrow[d] & B_0 \arrow[d] & I \arrow[l] \arrow[d, equal] \\
B_1 \times_I E_1 \arrow[r] & B_1 & I \arrow[l]
\end{tikzcd}$$

We established a free-forgetful adjunction $\boldsymbol{PolyFun}^{\mathrm{c}}(I) \rightleftarrows \boldsymbol{PolyMnd}(I)$.

Now we generalise this to the setting of variable types. Let $\boldsymbol{PolyEnd}$ denote the category whose objects are polynomial functors $I \leftarrow E \rightarrow B \rightarrow I$ and whose morphisms are diagrams

$$\begin{tikzcd}
I_0 \arrow[d, "\alpha"'] & E_0 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & B_0 \arrow[r] \arrow[d] & I_0 \arrow[d, "\alpha"] \\
I_1 & E_1 \arrow[l] \arrow[r] & B_1 \arrow[r] & I_1
\end{tikzcd}$$

We have established a Grothendieck fibration

$$\textbf{\textit{PolyEnd}} \to \textbf{\textit{Set}}$$

sending a polynomial endofunctor to its set of types. This is actually the fibration ***PolyFun*** $\to$ ***Set*** $\times$ ***Set*** pulled back along the diagonal map ***Set*** $\to$ ***Set*** $\times$ ***Set***.

To say this is a fibration means that for a fixed arrow $\alpha : \overline{I} \to I$ in ***Set***, we have a natural bijection

$$\textbf{\textit{PolyEnd}}_\alpha(P, Q) = \mathrm{Nat}(P, \alpha^\sharp Q)$$

Let ***PolyMnd*** denote the category whose objects are polynomial monads and whose morphisms are monad maps: these are diagrams like the previous one required to respect the monad structure. This is most easily expressed in the partial-composition viewpoint where it amounts to requiring that these two squares commute:

$$\begin{tikzcd}
B_0 \times_{I_0} E_0 \arrow[r] \arrow[d] & B_0 \arrow[d] & I_0 \arrow[l] \arrow[d] \\
B_1 \times_{I_1} E_1 \arrow[r] & B_1 & I_1 \arrow[l]
\end{tikzcd}$$

ALTERNATIVELY, WE SAY THAT A MORPHISM OF MONADS IS A DIAGRAM WHOSE VERTICAL PART IS A MAP OF MONADS IN THE SINGLE-SORT SETTING.

We should take advantage of the strictly extensional viewpoint here!

$$\begin{tikzcd}
\textbf{\textit{Set}}/I' \arrow[r, "M'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow"] & \textbf{\textit{Set}}/I' \arrow[d, "u_!"'] \\
\textbf{\textit{Set}}/I \arrow[r, "M"'] & \textbf{\textit{Set}}/I
\end{tikzcd}$$

We are implicitly referring to the double category $\textbf{\textit{PolyFun}}^{\mathrm{c}}$. If we restrict to a fixed I, then we find a monoidal category $\textbf{\textit{PolyFun}}^{\mathrm{c}}(I)$, and the monads are just monoids in here.

13.2.1 Proposition. *The end-point functor* ***PolyMnd*** $\to$ ***Set*** *is fibred. The cartesian lifts are the same as for the for* ***PolyEnd*** $\to$ ***Set****: namely: if Q is a monad, then $\alpha^\sharp Q$ is naturally a monad and the canonical map $\alpha^\sharp Q \to Q$ is a monad map.*

Proof. We show that $\alpha^\sharp Q$ is naturally a monad. Let the structure maps of Q be denoted

$$\mathsf{Id}_{\boldsymbol{Set}/I} \Rightarrow Q \overset{\mu}{\Leftarrow} QQ$$

Apply $\alpha^\sharp$:

$$\alpha_! \alpha^* \Rightarrow \alpha_! \cdot Q \cdot \alpha^*$$

Precomposing with the unit of the adjunction $\alpha_! \dashv \alpha^*$ we get the unit for the new monad. On the other hand the new composition law is given as the composite of the old composition law and the counit for the adjunction:

$$\alpha_! \cdot Q \cdot \alpha^* \cdot \alpha_! \cdot Q \cdot \alpha^* \Rightarrow \alpha_! \cdot Q \cdot Q \cdot \alpha^* \Rightarrow \alpha_! \cdot Q \cdot \alpha^*$$

It is very easy to check that this is again a monad.

If we agree to define monad maps as those whose vertical part is a monad map, then there is nothing more to prove here.

□

It follows immediately from this description that the forgetful functor ***PolyMnd*** $\to$ ***PolyEnd*** is fibred over ***Set*** and furthermore takes cartesian arrows to cartesian arrows (by construction).

13.2.2 Proposition. *The forgetful functor* ***PolyMnd*** $\to$ ***PolyEnd*** *has a left adjoint $P \mapsto \overline{\mathsf{P}}$, the free-monad functor. In other words, for each polynomial endofunctor P and each polynomial monad M, there is a natural bijection*

$$\boldsymbol{PolyEnd}(P, M) = \boldsymbol{PolyMnd}(\overline{\mathsf{P}}, M).$$

That the bijection is natural in P means that given $\alpha : Q \to P$ then the diagram

$$\begin{tikzcd}
\boldsymbol{PolyEnd}(P, M) \arrow[r, phantom, "="] \arrow[d, "\text{precomp. } \alpha"'] & \boldsymbol{PolyMnd}(\overline{\mathsf{P}}, M) \arrow[d, "\text{precomp. } F\alpha"] \\
\boldsymbol{PolyEnd}(Q, M) \arrow[r, phantom, "="] & \boldsymbol{PolyMnd}(\overline{\mathsf{Q}}, M)
\end{tikzcd}$$

commutes.

That the bijection is natural in M means that given monad map $\beta : M' \to M$ then the diagram

$$\begin{tikzcd}
\textbf{\textit{PolyEnd}}(P,M) \arrow[r, phantom, "="] & \textbf{\textit{PolyMnd}}(\overline{\mathsf{P}},M) \\
\textbf{\textit{PolyEnd}}(P,M') \arrow[u, "\text{postcomp. } \beta"] \arrow[r, phantom, "="] & \textbf{\textit{PolyMnd}}(\overline{\mathsf{P}},M') \arrow[u, "\text{postcomp. } F\beta"']
\end{tikzcd}$$

commutes.

Proof. The statement is that the 'insertion of generators' $P \Rightarrow \overline{\mathsf{P}}$ has a universal property with respect to general variable-sort polynomial morphisms, not just with respect to natural transformations. That is: For every monad Q and every morphism $P \to Q$, there is a unique monad map $\overline{\mathsf{P}} \to Q$ such that

$$\begin{tikzcd}
P \arrow[rr, phantom, "\Rightarrow"] \arrow[dr, "\phi"'] & & \overline{\mathsf{P}} \arrow[dl, "\exists!\text{monad map}"] \\
& Q &
\end{tikzcd}$$

Let α denote the endpoint map of ϕ. We know by the fibred property that the map ϕ factors through $\alpha^\sharp Q$, and we know by definition of monad map that the monad maps $\overline{\mathsf{P}} \to Q$ are in bijection with vertical monad maps $u : \overline{\mathsf{P}} \Rightarrow \alpha^\sharp Q$.

But inside the vertical category we have the universal property saying that there is only one such map u. (Some easy checks complete this argument.)

□

13.2.3 Monads in the category of arrows. There is also this viewpoint: consider the category of arrows $\textbf{\textit{Set}}^2$. Its objects are set maps $I_0 \to I_1$. Its morphisms are squares. It is a locally cartesian closed category. Consider polynomial functors in here. We can restrict to those for which $p = (p_0, p_1)$ is a cartesian square. Now in this category of polynomial functors we can look at monads. A monad in here will be a set map $I_0 \to I_1$

SEE SEPARATE NOTES FEB2007

relative

If now P is a monad, then there is natural monoidal structure on ***PolyFun***$^{\mathrm{c}}/P$: the tensor product of Q with R is the composite $Q \circ R$ with P structure given by $Q \circ R \to P \circ P \to P$.

WE CHECKED SOMEWHERE THAT COMPOSITION IS COMPATIBLE WITH base change!!!

Let ***PolyMnd*** denote the category of all polynomial monads. The arrows in this category are diagrams

$$\begin{tikzcd}
 & E \arrow[r] \arrow[dl] \arrow[dd] \arrow[dr, phantom, "\lrcorner", very near start] & B \arrow[dd, "\alpha"'] \arrow[dr] & \\
I \arrow[dd] & & & I \arrow[dd] \\
 & E' \arrow[r] \arrow[dl] & B' \arrow[dr] & \\
I' & & & I'
\end{tikzcd} \tag{13.1}$$

alpha

that respect the monad structure. This is most easily expressed in the partial-composition viewpoint where it amounts to requiring that these two squares commute:

$$\begin{tikzcd}
B \times_I E \arrow[r] \arrow[d] & B \arrow[d] & I \arrow[l] \arrow[d] \\
B' \times_{I'} E' \arrow[r] & B' & I' \arrow[l]
\end{tikzcd}$$

13.3 Coloured operads and generalised operads

13.4 P-spans, P-multicategories, and P-operads

sec:spans-ope

13.4.1 Spans. Let ***Span*** denote the bicategory of spans in ***Set***, as introduced in [15]. Under the interpretation of spans as linear polynomials (cf. Example **??**), composition of spans (resp. morphisms of spans) agrees with composition of polynomials (resp. morphisms of polynomial), so we can regard ***Span*** as a locally full sub-bicategory of ***Poly***$^{\mathrm{c}}$, and view polynomials as a natural 'non-linear' generalisation of spans.

It is a fundamental observation, due to Bénabou [15], that (small) categories are precisely monads in the bicategory of spans (in ***Set***). If P is a cartesian monad on a cartesian category $\mathscr{E}$, there is a notion of P-span, due to Burroni [24]: a P-span is a diagram $PD \leftarrow N \rightarrow C$. The P-spans are the arrows of a bicategory whose objects are those of $\mathscr{E}$. The monads in here are the P-multicategories in the sense of Leinster [75, 4.2.2]. Monads on the terminal object of $\mathscr{E}$ (if there is one) are the P-operads. The classical notion of multicategory and (nonsymmetric) operad can be recovered as special cases of this general definition by taking P to be the list monad on ***Set***. The category of P-operads is naturally equivalent to the category of monads over P (i.e. monads equipped with a monad map to P). This observation goes back to Kelly (see [58]). More generally, the category of P-multicategories is naturally equivalent to a category of monads with cartesian colax monad map to P (cf. Leinster [75, Prop. 6.2.3]).

This last result already involves base change subtleties, and it can be clarified by passing to a double-categorical viewpoint: the two categories are just categories of monads in two equivalent double categories, and not only are those two double categories equivalent, their constructions are essentially the same and amounts to nothing more than the intensive/extensive difference that is a central theme of this work. After establishing these results, we compare the notion of P-spans with that of polynomial functors over P, exploiting a direct intensional comparison.

13.4.2 Burroni spans. There is another notion of 'non-linear' span, namely the P-spans of Burroni [24], which is a notion relative to is a cartesian monad P. This section is dedicated to a systematic comparison between the two notions, yielding (for a fixed polynomial monad P) an equivalence of framed bicategories between Burroni P-spans and polynomials over P in the double-category sense. We show how the comparison can be performed directly at the level of diagrams by means of some pullback constructions. Considering monads in these categories, we find an equivalence between P-multicategories (also called coloured P-operads) and polynomial monads over P, in the double-category sense.

In this section, strength plays no essential role: everything is cartesian relative to a fixed P, eventually assumed to be polynomial and hence strong, and for all the cartesian natural transformations into P there is a unique way to equip the domain with a strength in such a way that the natural transformation becomes strong.

13.4.3 ***P*-spans.** We first need to recall some material on P-spans and their extension. To avoid clutter, and to place ourselves in the natural level of generality, we work in a cartesian closed category $\mathscr{C}$, and consider a fixed cartesian endofunctor $P : \mathscr{C} \to \mathscr{C}$. We shall later substitute $\boldsymbol{Set}/I$ for $\mathscr{C}$, and assume that P is a polynomial monad on $\boldsymbol{Set}/I$.

13.4.4 ***P*-spans.** By definition, a *P-span* is a diagram in $\mathscr{C}$ of the form

$$P(D) \xleftarrow{d} N \xrightarrow{c} C\,, \tag{13.2}$$

equ:pspan

A *morphism of P-spans* is a diagram like

$$\begin{tikzcd}
P(D') \arrow[d, "P(f)"'] & N' \arrow[l, "d'"'] \arrow[r, "c"] \arrow[d, "g"] & C' \arrow[d, "h"] \\
P(D) & N \arrow[l, "d"] \arrow[r, "c"'] & C\,,
\end{tikzcd} \tag{13.3}$$

equ:pspanmorph

We write P-***Span*** for the category of P-spans and P-span morphisms in $\mathscr{C}$.

13.4.5 Extension of *P*-spans. Let $\boldsymbol{Slice}^{\mathrm{c}}_{\mathscr{C}}$ denote the category whose objects are cartesian functors between slices of $\mathscr{C}$ and whose arrows are diagrams of the form

$$\begin{tikzcd}
\mathscr{C}/D' \arrow[r, "Q'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow\psi"] & \mathscr{C}/C' \arrow[d, "v_!"] \\
\mathscr{C}/D \arrow[r, "Q"'] & \mathscr{C}/C\,,
\end{tikzcd} \tag{13.4}$$

psi

for $u : D' \to D$ and $v : C' \to C$ in $\mathscr{C}$, and ψ a cartesian natural transformation. Under the identification $\mathscr{C} = \mathscr{C}/1$, we can consider P as an object of $\boldsymbol{Slice}^{\mathrm{c}}_{\mathscr{C}}$, so it makes sense to consider the slice category $\boldsymbol{Slice}^{\mathrm{c}}_{\mathscr{C}}/P$: its objects are the cartesian functors $Q : \mathscr{C}/D \to \mathscr{C}/C$ equipped with a cartesian natural transformation

$$\begin{tikzcd}
\mathscr{C}/D \arrow[r, "Q"] \arrow[d] \arrow[dr, phantom, "\Swarrow\phi"] & \mathscr{C}/C \arrow[d] \\
\mathscr{C} \arrow[r, "P"'] & \mathscr{C}\,.
\end{tikzcd} \tag{13.5}$$

equ:cartsq

Given two such objects

$$\begin{tikzcd}
\mathscr{C}/D' \arrow[r, "Q'"] \arrow[d] \arrow[dr, phantom, "\Swarrow \phi'"] & \mathscr{C}/C' \arrow[d] & & \mathscr{C}/D \arrow[r, "Q"] \arrow[d] \arrow[dr, phantom, "\Swarrow \phi"] & \mathscr{C}/C \arrow[d] \\
\mathscr{C} \arrow[r, "P"'] & \mathscr{C} & & \mathscr{C} \arrow[r, "P"'] & \mathscr{C} ,
\end{tikzcd}$$

a morphism from the former to the latter is a square ψ like (13.4) satisfying

$$\begin{tikzcd}
\mathscr{C}/D' \arrow[r, "Q'"] \arrow[d] \arrow[dr, phantom, "\Swarrow \phi'"] & \mathscr{C}/C' \arrow[d] \\
\mathscr{C} \arrow[r, "P"'] & \mathscr{C}
\end{tikzcd}
\quad = \quad
\begin{tikzcd}
\mathscr{C}/D' \arrow[r, "Q'"] \arrow[d, "u_!"'] \arrow[dr, phantom, "\Swarrow \psi"] & \mathscr{C}/C' \arrow[d, "v_!"] \\
\mathscr{C}/D \arrow[r, "Q"] \arrow[d] \arrow[dr, phantom, "\Swarrow \phi"] & \mathscr{C}/C \arrow[d] \\
\mathscr{C} \arrow[r, "P"'] & \mathscr{C} .
\end{tikzcd}$$

We now construct a functor Ext : P-$\textbf{\textit{Span}}_{\mathscr{C}} \to \textbf{\textit{Slice}}^{\mathrm{c}}_{\mathscr{C}}/P$. Its action on objects is defined by mapping a P-span $PD \xleftarrow{d} N \xrightarrow{c} C$ to the diagram

$$\begin{tikzcd}
\mathscr{C}/D \arrow[r, "P_{/D}"] \arrow[d] & \mathscr{C}/PD \arrow[r, "d*"] \arrow[d] \arrow[dr, phantom, "\Swarrow"] & \mathscr{C}/N \arrow[r, "c_!"] \arrow[d] & \mathscr{C}/C \arrow[d] \\
\mathscr{C} \arrow[r, "P"'] & \mathscr{C} \arrow[r, equal] & \mathscr{C} \arrow[r, equal] & \mathscr{C} .
\end{tikzcd}$$

Here $P_{/D} : \mathscr{C}/D \to \mathscr{C}/PD$ sends $f : X \to D$ to $Pf : PX \to PD$, and the outer squares are commutative. The middle square is essentially given by the counit of the adjunction $d_! \dashv d*$, and is therefore a cartesian natural transformation. More precisely, it is the mate [60] of the commutative square

$$\begin{tikzcd}
\mathscr{C}/PD \arrow[d] & \mathscr{C}/N \arrow[l, "d_!"'] \arrow[d] \\
\mathscr{C} \arrow[r, equal] & \mathscr{C} .
\end{tikzcd}$$

The action of the functor Ext : P-$\textbf{\textit{Span}}_{\mathscr{C}} \to \textbf{\textit{Slice}}^{\mathrm{c}}_{\mathscr{C}}/P$ on morphisms is defined by mapping a diagram like (13.3) to the natural transformation

$$\begin{tikzcd}
\mathscr{C}/D' \arrow[r, "P_{/D'}"] \arrow[d, "f_!"'] & \mathscr{C}/PD' \arrow[r, "d'^*"] \arrow[d, "Pf_!"'] \arrow[dr, phantom, "\swarrow"] & \mathscr{C}/N' \arrow[r, "c'_!"] \arrow[d, "g_!"] & \mathscr{C}/C' \arrow[d, "h_!"] \\
\mathscr{C}/D \arrow[r, "P_{/D}"'] & \mathscr{C}/PD \arrow[r, "d^*"'] & \mathscr{C}/N \arrow[r, "c_!"'] & \mathscr{C}/C ,
\end{tikzcd}$$

together with the structure maps to P. The outer squares are just commutative and the middle square (again cartesian) is the mate of the identity 2-cell

$$\begin{tikzcd}
\mathscr{C}/PD' \arrow[d, "(Pf)_!"'] & \mathscr{C}/N' \arrow[l, "d'_!"'] \arrow[d, "g_!"] \\
\mathscr{C}/PD & \mathscr{C}/N . \arrow[l, "d_!"]
\end{tikzcd}$$

thm:functofspan

13.4.6 Proposition. *The functor* Ext : P-***Span***$_{\mathscr{C}} \to$ ***Slice***$^c_{\mathscr{C}}/P$ *is an equivalence of categories.*

Proof. The quasi-inverse is defined by mapping

$$\begin{tikzcd}
\mathscr{C}/D \arrow[r, "Q"] \arrow[d] \arrow[dr, phantom, "\swarrow\phi"] & \mathscr{C}/C \arrow[d] \\
\mathscr{C} \arrow[r, "P"'] & \mathscr{C}
\end{tikzcd}$$

to the P-span

$$PD \xleftarrow{\phi_D} QD \xrightarrow{Q(1_D)} C .$$

The verification of the details is straightforward. □

13.4.7 Change of shape. Given a cartesian natural transformation $\theta : P \Rightarrow P'$, there is a shape-change functor

$$\begin{array}{ccc}
P\text{-}\textbf{\textit{Span}}_{\mathscr{C}} & \longrightarrow & P'\text{-}\textbf{\textit{Span}}_{\mathscr{C}} \\
[PD \leftarrow N \rightarrow C] & \longmapsto & [P'D \overset{\theta_D}{\leftarrow} PD \leftarrow N \rightarrow C] .
\end{array}$$

We also have the functor

$$\begin{array}{rcl} \boldsymbol{Slice}^{\mathrm{c}}_{\mathscr{C}}/P & \longrightarrow & \boldsymbol{Slice}^{\mathrm{c}}_{\mathscr{C}}/P' \\ [Q \Rightarrow P] & \longmapsto & [Q \Rightarrow P \overset{\theta}{\Rightarrow} P']\,. \end{array}$$

13.4.8 Lemma. *The equivalence* Ext *of Proposition 13.4.6 is compatible with change of shape, in the sense that the following diagram commutes:*

$$\begin{tikzcd} P\text{-}\boldsymbol{Span}_{\mathscr{C}} \arrow[r, "\mathrm{Ext}"] \arrow[d] & \boldsymbol{Slice}^{c}_{\mathscr{C}}/P \arrow[d] \\ P'\text{-}\boldsymbol{Span}_{\mathscr{C}} \arrow[r, "\mathrm{Ext}"'] & \boldsymbol{Slice}^{c}_{\mathscr{C}}/P'\,. \end{tikzcd}$$

Proof. The claim amounts to checking

$$\theta^*_D \circ P'_{/D} = P_{/D}$$

which follows from the assumption that θ is cartesian. □

In detail, given $X \to D$, apply P' to get $P'X \to P'D$. Then pullback along θ_D, which since θ is cartesian gives $PX \to PD$.

13.4.9 More general change of shape, not needed here. In fact more generally there is shape-change along maps like

$$\begin{tikzcd} \mathscr{E}/I' \arrow[r, "T'"] \arrow[d, "k_!"'] & \mathscr{E}/I' \arrow[d, "k_!"] \arrow[dl, phantom, "\swarrow\,\theta"] \\ \mathscr{E}/I \arrow[r, "T"'] & \mathscr{E}/I \end{tikzcd}$$

Then a T'-span $T'D \leftarrow N \rightarrow C$ on $\mathscr{E}/I'$ is sent to the T-span on $\mathscr{E}/I$ given by:

$$Tk_!\,D \leftarrow k_!\,T'D \leftarrow k_!\,N \rightarrow k_!\,C$$

Similarly, the change functor from $\boldsymbol{Cart}/T'$ to $\boldsymbol{Cart}/T$ is just given by pasting with θ.

13.4.10 Composition of P-spans and composition of their extensions. We now assume that P is a cartesian monad, so we have two natural transformations $\eta : 1 \Rightarrow P$ and $\mu : P \circ P \Rightarrow P$ at our disposal for shape-change.

As is well known [75], this allows us to define horizontal composition of P-spans: given composable P-spans

$$\begin{tikzcd}
& N \arrow[dl, "d"'] \arrow[dr, "c"] & & & U \arrow[dl, "s"'] \arrow[dr, "t"] & \\
PD & & C & PC & & B,
\end{tikzcd}$$

we define their composite P-span by applying P to the first P-span, performing a pullback, and using the multiplication map:

$$\begin{tikzcd}
& & & PN \times_{PC} U \arrow[dl] \arrow[dr] & & \\
& & PN \arrow[dl, "Pd"'] \arrow[dr, "Pc"] & & U \arrow[dl, "s"'] \arrow[dr, "t"] & \\
& PPD \arrow[dl, "\mu_D"'] & & PC & & B \\
PD & & & & &
\end{tikzcd} \tag{13.6}$$

Pspan-comp

13.4.11 Lemma. *This composition is associative (up to those coherences) if T preserves pullbacks and the natural transformation $m : TT \to T$ is cartesian.*

13.4.12 Lemma. *The unit arrow is the diagram*

$$\begin{tikzcd}
& I \arrow[dl, "u"'] \arrow[dr] & \\
T(I) & & I
\end{tikzcd}$$

Proof. It is immediate to check that this works as a unit from the left. To see that it works as unit from the right, we need to assume that T is a cartesian monad, precisely we need that the naturality square for the unit is cartesian. Here is the diagram for composing with id_J on the right:

$$\begin{tikzcd}
 & & A \arrow[dl, "u_A"'] \arrow[dr] & & \\
 & T(A) \arrow[dl] \arrow[dr] \arrow[r, phantom, "\text{p.b.}"] & & J \arrow[dl, "u_J"] \arrow[dr] & \\
T(T(I)) \arrow[d, "m_I"'] & & T(J) & & J \\
T(I) & & & &
\end{tikzcd}$$

The fact that the top square is cartesian is precisely because it is the unit naturality square. To see that the long composite on the left is equal to the original map $f : A \to T(I)$ is a consequence of the fact that the unit is really a unit for the composition functor M. Precisely we have this commutative square

$$\begin{tikzcd}
TA \arrow[d, "Tf"'] & A \arrow[l, "u_A"'] \arrow[d, "f"] \\
TTI \arrow[d, "m_I"'] & TI \arrow[l, "u_{TI}"] \arrow[dl, "\mathrm{id}"] \\
TI &
\end{tikzcd}$$

□

Associativity of the composition law (up to coherent isomorphism) depends on that fact that P preserves pullbacks and that μ is cartesian. It further follows from the fact that η is cartesian that for each D the P-span

$$PD \xleftarrow{\eta_D} D \xrightarrow{1_D} D$$

is the identity P-spans for the composition law (up to coherent isomorphisms). It is clear that these constructions are functorial in vertical maps between P-spans, yielding altogether a double category of P-spans, denoted P-$\boldsymbol{Span}_{\mathscr{C}}$: the objects and vertical morphisms are those of $\mathscr{C}$, the horizontal arrows are the P-spans, and the squares are diagrams like (13.3).

13.4.13 Double category of cartesian functors. We also have a double-category structure on $\textbf{\textit{Slice}}^{\mathrm{c}}_{\mathscr{C}}/P$: the horizontal composite of $Q \Rightarrow P$ with $R \Rightarrow P$ is $R \circ Q \Rightarrow P \circ P \Rightarrow P$, and the horizontal identity arrow is $\mathsf{Id} \Rightarrow P$. Let us verify that the extension of a horizontal composite is isomorphic to the composite of the extensions: in the diagram

$$\begin{tikzcd}
 & & & \mathscr{C}/C \arrow[dr, "P_{/C}"] & & & \\
 & & \mathscr{C}/N \arrow[ur, "c_!"] \arrow[dr, "P_{/N}"'] & & \mathscr{C}/PC \arrow[dr, "s*"] & & \\
 & \mathscr{C}/PD \arrow[ur, "d*"] \arrow[dr, "P_{/PD}"] & & \mathscr{C}/PN \arrow[ur, "(Pc)_!"] \arrow[dr] & \text{B.C.} & \mathscr{C}/U \arrow[dr, "t_!"] & \\
\mathscr{C}/D \arrow[ur, "P_{/D}"] \arrow[rr, "(PP)_{/D}"'] & & \mathscr{C}/PPD \arrow[ur, "(Pd)*"'] & & \mathscr{C}/PN \times_{PC} U \arrow[ur] & & \mathscr{C}/B
\end{tikzcd}$$

the top path is the composite of the extension functors, and the bottom path is the extension of the composite span. The square marked B.C. is the Beck-Chevalley isomorphism for the cartesian square (13.6), and the other squares, as well as the triangle, are clearly commutative. The following proposition now follows from Proposition 13.4.6.

13.4.14 Proposition. *The functor* Ext : P-$\textbf{\textit{Span}}_{\mathscr{C}} \longrightarrow \textbf{\textit{Slice}}^{c}_{\mathscr{C}}/P$ *is an equivalence of double categories, in fact an equivalence of framed bicategories.*

We just owe to make explicit how the double category of P-spans is a framed bicategory: to each vertical map $u : D' \to D$ we associate the P-span

$$PD' \xleftarrow{\eta_{D'}} D' \xrightarrow{u} D\,.$$

This is a left adjoint; its right adjoint is the P-span

$$PD \xleftarrow{\eta_D \circ u} D' \xrightarrow{=} D'$$

as follows by noting that their extensions are respectively $u_!$ and u^*. For this the important fact is that η is cartesian. With this observation it is clear that the equivalence is framed.

13.4.15 Remark. Another useful formula, although it is not so precise what the types are concerned: recall that for any cartesian natural transformation $\theta : S \Rightarrow T$ on a cartesian category with 1, we have

$$S(X) = T(X) \times_{T(1)} S(1)$$

saying that S is determined by its value on 1 and by T. In a variable-sorts setting, given a cartesian natural transformation

$$\begin{tikzcd}
\mathscr{E}/D \arrow[r, "Q"] \arrow[d, "\alpha_!"'] & \mathscr{E}/C \arrow[d, "\beta_!"] \arrow[dl, phantom, "\swarrow \theta"] \\
\mathscr{E}/I \arrow[r, "P"'] & \mathscr{E}/J
\end{tikzcd}$$

(recall that this is like $Q \Rightarrow \beta^* P \alpha_!$) the formula reads

$$\beta_! Q(X) = P\alpha_!(X) \times_{P\alpha_!(D)} \beta_! Q(D)$$

With this type-sloppy notation, it is clear that Q and the functor defined by the associated span are the same: Q is $X \mapsto P(X) \times_{P(D)} N$, and the functor defined by the associated span is $X \mapsto P(X) \mapsto P(X) \times_{P(D)} N$. That's the same formula.

[Taking the mate of this 2-cell, we get also

$$Q\alpha^*(Y) = \beta^* P(Y) \times_{\beta^* P(I)} Q\alpha^*(I)$$

but we shall not need this one.]

The $\beta_! Q$ should be thought of as $q_* z^*$, while the fibre product on the right-hand side is to apply d^* to $P\alpha_!$ getting altogether $d^* f_* \varepsilon^* a^*$. Seeing that these are equal is just to use $q_* k^* = d^* f_*$.

13.4.16 Polynomial functors over P. ADJUST THIS TO WHAT WAS ALREADY SAID ABOUT THIS Fix a polynomial endofunctor $P : \textbf{\textit{Set}}/I \to \textbf{\textit{Set}}/I$ given by $I \leftarrow E \to B \to I$. Let $\textbf{\textit{Poly}}/P$ denote the framed bicategory sliced over P. Its objects and vertical arrows are those of $\textbf{\textit{Set}}/I$; its horizontal arrows are polynomial functors $\textbf{\textit{Set}}/D \to \textbf{\textit{Set}}/C$ over P, like

$$\begin{tikzcd}
& D \arrow[d] & U \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & V \arrow[r] \arrow[d] & C \arrow[d] \\
P: & I & E \arrow[l] \arrow[r] & B \arrow[r] & I.
\end{tikzcd}$$

and its 2-cells are

$$\begin{tikzcd}
& D' \arrow[d] & U' \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & V' \arrow[r] \arrow[d] & C' \arrow[d] \\
& D \arrow[d] & U \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & V \arrow[r] \arrow[d] & C \arrow[d] \\
P: & I & E \arrow[l] \arrow[r] & B \arrow[r] & I.
\end{tikzcd} \tag{13.7}$$

$\boxed{\texttt{D'D}}$

(i.e. compatible morphisms of polynomial functors compatible with the structure map to P).

It is observed in [63] (see a later chapter of these notes) that the subcategory $\textbf{\textit{PolyEnd}}/P \subset \textbf{\textit{Poly}}/P$ is equivalent to a category of sheaves (on a certain site of P-trees)—at least in the case $\mathscr{E} = \textbf{\textit{Set}}$. We do not know how to generalise that construction to non-endo polynomial functors over P.

13.4.17 Theorem. *There is an equivalence of double categories between this one and the double category of P-spans.*

This is clear since both types of diagrams represent the same functors.

We shall describe a direct and very explicit back-and-forth between the two types of diagrams, without reference to the functors they represent. (However, in order to prove that these constructions constitute an equivalence of framed bicategories, we shall take advantage of the fact that these things represent functors.)

13.4.18 Polynomial functors over a base. We now specialise to the case of interest, where $\mathscr{C} = \textbf{\textit{Set}}/I$ and P is a polynomial monad on $\textbf{\textit{Set}}/I$, represented by

$$I \leftarrow B \rightarrow A \rightarrow I\,.$$

Since now all the maps involved in the P-spans are over I, a P-span can be interpreted as a commutative diagram

$$\begin{tikzcd}
& N \arrow[ld, "d"'] \arrow[rd, "c"] & \\
PD \arrow[rd] & & C \arrow[ld] \\
& I\,. &
\end{tikzcd}$$

If C is an object of $\mathscr{C}$, i.e. a map in $\textbf{\textit{Set}}$ with codomain I, we shall write C also for its domain, and we have a natural identification of slices $\mathscr{C}/C \simeq \textbf{\textit{Set}}/C$. That $P : \textbf{\textit{Set}}/I \rightarrow \textbf{\textit{Set}}/I$ is a polynomial monad, means thanks to Lemma 10.1.6, that all objects in $\textbf{\textit{Slice}}^{\mathrm{c}}_{\textbf{\textit{Set}}/I}/P$ are polynomial again, so $\textbf{\textit{Slice}}^{\mathrm{c}}_{\textbf{\textit{Set}}/I}/P \cong \textbf{\textit{Poly}}^{\mathrm{c}}/P$, the category of polynomials cartesian over P in the double-category sense. In conclusion:

thm:PSpan=Poly/P

13.4.19 Proposition. *The functor* Ext : P-$\textbf{\textit{Span}}_{\textbf{\textit{Set}}/I} \rightarrow \textbf{\textit{Poly}}^{c}/P$ *is an equivalence of framed bicategories.*

□

13.4.20 Comparison of hom categories. It is a natural question whether there is a direct comparison between P-spans and polynomials over P, without reference to their extensions. This is indeed the case, as we now proceed to establish, exploiting the framed structure. Given a polynomial over P, like

$$\begin{tikzcd}
Q: & D \arrow[d, "u"'] & M \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & N \arrow[r] \arrow[d] & C \arrow[d] \\
P: & I & B \arrow[l] \arrow[r] & A \arrow[r] & I\,.
\end{tikzcd}$$

Consider the canonical factorisation of this morphism through the source lift of P along u (cf. 11.1.13):

$$\begin{tikzcd}
Q: & D \arrow[d, equal] & M \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & N \arrow[r, "c"] \arrow[d, "d"'] & C \arrow[d] \\
(u,\mathrm{id}\,)^\sharp P: & D \arrow[d, equal] & \cdot \arrow[l] \arrow[r, "f"] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & PD \arrow[r] \arrow[dd] & I \arrow[dd, equal] \\
& D \arrow[d, "u"'] & \cdot \arrow[l] \arrow[d] \arrow[dl, phantom, "\llcorner", very near start] & & \\
P: & I & B \arrow[l] \arrow[r] & A \arrow[r] & I\,.
\end{tikzcd} \tag{13.8}$$

QP-poly-span

Now we just read off the associated P-span:

$$\begin{tikzcd}
N \arrow[r, "c"] \arrow[d, "d"'] & C \arrow[d] \\
PD \arrow[r] & I\,.
\end{tikzcd}$$

Conversely, given such a P-span, place it on top of the rightmost leg of $P \circ \alpha_! = (u, \mathrm{id}\,)^\sharp P$ (the middle row of the diagram, which depends only on α and P), and let M be the pullback of $N \to P(D)$ along the arrow labelled f. It is easy to see that these constructions are functorial, yielding an equivalence of hom categories $\boldsymbol{Poly}^{\mathrm{c}}(D,C)/P \simeq P\text{-}\boldsymbol{Span}_{\mathbf{Set}/I}(D,C)$.

13.4.21 Example. Endo-P-spans $PC \leftarrow N \rightarrow C$, that is, polynomial endofunctors over P, are called *C-coloured P-collections*. If furthermore $C = I$ we simply call them *P-collections*. These are just polynomial endofunctors

$Q : \boldsymbol{Set}/I \to \boldsymbol{Set}/I$ equipped with a cartesian natural transformation to P. This category is itself a slice of $\boldsymbol{Set}$: it is easy to see that the functor

$$\begin{aligned} \boldsymbol{Poly}^{\mathrm{c}}(I,I)/P &\longrightarrow \boldsymbol{Set}/P1 \\ Q &\longmapsto [Q1 \to P1] \end{aligned}$$

is an equivalence of categories.

13.4.22 Multicategories. Burroni [24], Leinster [75], and Hermida [48] define P-multicategories (also called coloured P-operads) as monads in the bicategory of P-spans. P-multicategories are also monads in the double category of P-spans — this description also provides the P-multifunctors as (oplax) cartesian monad maps. P-multicategories based at the terminal object in $\boldsymbol{Set}/I$ are called P-operads. If the base monad P is a polynomial monad, the equivalence of Proposition 13.4.19 induces an equivalence of the categories of monads, as summarised in the corollary below.

In the classical example, $\boldsymbol{Set}$ is $\boldsymbol{Set}$ and P is the free-monoid monad M of Example 1.2.8. In this case, M-multicategories are the classical multicategories of Lambek [69], which are also called coloured nonsymmetric operads. In the one-object case, M-operads are the plain (nonsymmetric) operads. The other standard example is taking P to be the identity monad on $\boldsymbol{Set}$. Then P-multicategories are just small categories and P-operads are just monoids. Hence small categories are essentially polynomial monads on some slice $\boldsymbol{Set}/C$ with an oplax cartesian double-categorical monad map to Id, and monoids are essentially polynomial monads on $\boldsymbol{Set}$ with a cartesian monad map to Id. In summary, we have the following result.

Definition. Given a cartesian monad $T : \boldsymbol{Set} \to \boldsymbol{Set}$. For each set I there is a monoidal category (the vertex bicategory) of T-spans from I to I. The monoidal operation is $\circ$. A *T-multicategory* on I is by definition a monoid in this monoidal category. In other words, it is an endo T-span

$$\begin{tikzcd} & A \arrow[dl] \arrow[dr] & \\ TI & & I \end{tikzcd}$$

A *T-operad* is an T-multicategory on the object $1 \in \boldsymbol{Set}$.

13.4.23 Classical multicategories and classical operads. When T is the free-monoid monad, then the corresponding notion of multicategory is the classical notion of Lambek. An arrow $a \in A$ goes from a sequence of elements $(x_1, \ldots, x_n) \in I^*$ to a single element $y \in I\ldots$

When furthermore I is the singleton set, then we recover the classical notion of operad (nonsymmetric). If the monad is the free-monoid monad, then we get the classical notion of operad. This is because $M(1) = \mathbb{N}$. Then

$$\begin{tikzcd}
& A \arrow[dl] \arrow[dr] & \\
M(1) = \mathbb{N} & & 1
\end{tikzcd}$$

is just the collection $A \to \mathbb{N}$ defining a classical operad.

13.4.24 Categories and monoids. Let E denote the identity endofunctor $\textbf{\textit{Set}} \to \textbf{\textit{Set}}$. An E-multicategory on a set S is nothing but a plain category with object set S. Indeed, then an E-span is nothing but a plain span, and the statement reduces to the fact that a monoid in the monoidal category of endospans is the same thing as a category. An E-operad is nothing but a monoid.

In conclusion, every cartesian monad $T : \mathscr{E} \to \mathscr{E}$ gives rise to a notion of T-operad. there is a monoidal category of T-collections (or T-endospans): objects are endospans in $\mathscr{E}$:

$$\begin{tikzcd}
& X \arrow[dl] \arrow[dr] & \\
T(1) & & 1
\end{tikzcd}$$

Since 1 is terminal, this is just $\mathscr{E}/T1$.

13.4.25 Theorem. *(Essentially Leinster [75], Cor. 6.2.4.) There is a natural equivalence of monoidal categories between* $\textbf{\textit{Poly}}/T$ *and* $T - \textbf{\textit{Coll}}$.

$\textbf{\textit{Poly}}/T$ is the category of polynomial functors in $\mathscr{E}$ with a cartesian map to T. Since T is a monad, this category acquires a tensor product, see below.

Coloured operads

C-coloured operads are monoids in the category of M-endospans based on C instead of the terminal object. These are spans

$$\begin{tikzcd}
& X \arrow[dl] \arrow[dr] & \\
M(C) & & C
\end{tikzcd}$$

with monoid structure. This is $\mathbf{Set}/(MC \times C)$ rather than just $\mathbf{Set}/MC$.

13.4.26 Coloured operads. A *coloured operad*, also called a *multicategory*, consists of a pair (P, C) where C is a set whose elements are called *colours*, and P is a collection of sets indexed by $(n+1)$-tuples of colours, $n \in \mathbb{N}$. So for each n and for each $(c_1, \ldots, c_n; c)$ there is a set

$$P(c_1, \ldots, c_n; c)$$

of *operations* of input colours $(c_1, \ldots, c_n)$ and output colour c. These sets together have operad structure, so that you can substitute n operations into a given n-ary operation provided the colours match. This composition law is required to be associative and unital. Note that there is a unit operation for each colour, $1_c \in P(c; c)$.

More stuff about this. Reference to Berger–Moerdijk [18].

Algebras for a coloured operad.

13.4.27 Burroni version. A I-coloured operad is just a monoid in the monoidal category whose objects are I-based M-endospans:

$$\begin{tikzcd}
& X \arrow[dl] \arrow[dr] & \\
M(I) & & I
\end{tikzcd}$$

The set X is the total set of operations. The map to $M(I)$ sends each operation to its list of input colours, and the map to I returns the output colour.

Polynomial monads and coloured operads

13.4.28 Theorem. *A coloured operad is essentially the same thing as a polynomial monad over M:*

$$\begin{tikzcd} I \arrow[d] & X' \arrow[l] \arrow[r] \arrow[d] & X \arrow[r] \arrow[d] & I \arrow[d] \\ 1 & \mathbb{N}' \arrow[l] \arrow[r] & \mathbb{N} \arrow[r] & 1 \end{tikzcd}$$

Here $X' := \mathbb{N}' \times_{\mathbb{N}} X$ is the set of operations with one input marked, and the map $X' \to I$ returns the colour of that input slot.

13.4.29 Change of colours. If (Q, D) is a coloured operad and $\alpha : C \to D$ is a map of sets, then we can construct a C-coloured operad $\alpha^* Q$ where

$$\alpha^* Q(c_1, \ldots, c_n; c) := Q(\alpha(c_1), \ldots, \alpha(c_n); \alpha(c)).$$

Change-of-colours for coloured operads corresponds precisely to base change in the theory of polynomial functors.

13.4.30 Corollary. *There are natural equivalences of categories*

$$\begin{array}{rr} P\textbf{-}\boldsymbol{Multicat} \simeq \boldsymbol{PolyMnd}/P & P\textbf{-}\boldsymbol{Operad} \simeq \boldsymbol{PolyMnd}(1)/P \\ \boldsymbol{Multicat} \simeq \boldsymbol{PolyMnd}/M & \boldsymbol{PlainOperad} \simeq \boldsymbol{PolyMnd}(1)/M \\ \boldsymbol{Cat} \simeq \boldsymbol{PolyMnd}/\mathsf{Id} & \boldsymbol{Monoid} \simeq \boldsymbol{PolyMnd}(1)/\mathsf{Id}\,. \end{array}$$

13.4.31 Examples. The double category of polynomials is very convenient for reasoning with P-multicategories. The role of the base monad P for P-multicategories is to specify a profile for the operations. This involves specifying the shape of the input data, and it may also involve type constraints on input and output. In the classical case of $P = M$, the fibres of $\mathbb{N}' \to \mathbb{N}$ (Example 1.2.8) are finite ordinals, expressing the fact that inputs to an operation in a classical multicategory must be given as a finite list of objects. In this case there are no type constraints imposed by P on the operations.

For a more complicated example, let $P : \boldsymbol{Set}/\mathbb{N} \to \boldsymbol{Set}/\mathbb{N}$ be the free-plain-operad monad, which takes a collection (i.e. an object in $\boldsymbol{Set}/\mathbb{N}$) and returns the free plain operad on it [75, p.135, p.145, p.155]. This monad is polynomial (cf. [64]): it is represented by

$$\begin{tikzcd} \mathbb{N} & \mathrm{Tr}^\bullet \arrow[l, "s"'] \arrow[r, "p"] & \mathrm{Tr} \arrow[r, "t"] & \mathbb{N}, \end{tikzcd}$$

where Tr denotes the set of (isomorphism classes of) finite planar rooted trees, and $\mathrm{Tr}^\bullet$ denotes the set of (isomorphism classes of) finite planar rooted trees with a marked node. The map s returns the number of input edges of the marked node; the map p forgets the mark, and t returns the number of leaves. A P-multicategory Q has a set of objects and a set of operations. Each operation has its input slots organised as the set of nodes of some planar rooted tree, since this is how the p-fibres look like. Furthermore, there are type constraints: each object of Q must be typed in $\mathbb{N}$, via a number that we shall call the *degree* of the object, and a compatibility is required between the typing of operations and the typing of objects. Namely, the degree of the output object of an operation must equal the total number of leaves of the tree whose nodes index the input, and the degree of the object associated to a particular input slot must equal the number of incoming edges of the corresponding node in the tree. All this is displayed with clarity by the fact that Q is given by a diagram

$$\begin{tikzcd} Q: & D \arrow[d, "\alpha"'] & M \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner" very near start] & N \arrow[r] \arrow[d, "\beta"] & D \arrow[d, "\alpha"] \\ P: & \mathbb{N} & \mathrm{Tr}^\bullet \arrow[l] \arrow[r] & \mathrm{Tr} \arrow[r] & \mathbb{N} \end{tikzcd}$$

The typing of the operations is concisely given by the map β, and the organisation of the inputs in terms of the fibres of the middle map of P is just the cartesian condition on the middle square. The typing of objects is encoded by α and the compatibility conditions, somewhat tedious to formulate in prose, are nothing but commutativity of the outer squares.

Finite planar rooted trees can be seen as M-trees, where M : ***Set*** → ***Set*** is the free-monoid monad (1.2.8).

Chapter 14

Trees (2)

Ch:trees

This chapter is mostly copied verbatim from the paper *Polynomial functors and trees* [63].

In this chapter we construct a bigger category of trees than the one we had.

We found the following explicit construction of the free monad on P: if P is given by the diagram $A \leftarrow M \to N \to A$, then the free monad on P is given by

$$A \leftarrow \mathrm{tr}'(\mathsf{P}) \to \mathrm{tr}(\mathsf{P}) \to A$$

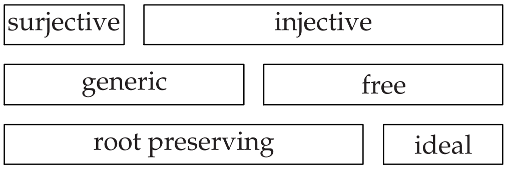
where $\mathrm{tr}'(\mathsf{P})$ denotes the set of isomorphism classes of P-trees with a marked leaf (14.1.2). The monad structure is given by grafting P-trees. We are particularly interested in free monads on trees. Since maps between trees are embeddings, the free monad on a tree $\mathsf{T} = (A \leftarrow M \to N \to A)$ is given by

$$A \leftarrow \mathrm{sub}'(\mathsf{T}) \to \mathrm{sub}(\mathsf{T}) \to A$$

(where $\mathrm{sub}(\mathsf{T})$ (resp. $\mathrm{sub}'(\mathsf{T})$) denotes the set of subtrees of T (resp. subtrees with a marked leaf)).

We now define ***Tree*** to be the category whose objects are trees and whose arrows are maps between their free monads (14.2.1). In other words, ***Tree*** is a full subcategory of ***PolyMnd*** (the category of polynomial monads): it is the Kleisli category of ***TEmb*** with respect to the free-monad monad on ***PolyEnd***. (It is shown that any map in ***PolyEnd*** between free monads on trees is a monad map (14.2.2).) In explicit terms, morphisms send edges to edges and subtrees to subtrees. The category ***Tree*** is equivalent to the category Ω of Moerdijk and Weiss [83], whose presheaves are called dendroidal sets. Its construction in terms of polynomial functors reveals important properties analogous to properties of Δ. In fact, Δ is equivalent to the full subcategory in ***Tree*** consisting of the linear trees.

The main intrinsic features of the category ***Tree*** are expressed in terms of three factorisation systems: ***Tree*** is shown to have has surjective/injective factorisation, generic/free factorisation, as well as root-preserving/ideal-embedding factorisation. The generic maps are precisely the boundary-preserving maps, and the generic/free factorisation system plays an important role in the second part of the paper. The compatibilities between the three factorisation systems are summarised in this figure:



As suggested by the figure, every arrow factors essentially uniquely as a surjection followed by a generic injection, followed by a free root-preserving map followed by an ideal embedding. Explicit descriptions are derived for each of the four classes of maps: the surjections consist in deleting unary nodes, the generic injections are node refinements (and of course the free maps are the tree embeddings). The ideal embeddings are those corresponding to subtrees containing all descendants—this is the notion of subtree most relevant to computer science and linguistics.

The subcategory of generic tree maps is opposite to the category of trees studied by Leinster [75, §7.3]. On Leinster's side of the duality, tree maps can be described in terms of set maps between the sets of nodes. On our side of the duality, tree maps are described in terms of set maps between the sets of edges. The category of generic injections is roughly the opposite of the category of trees studied[1] by Ginzburg and Kapranov in their seminal paper [41]; the difference is that they exclude all trees with nullary nodes. In fact, most of the time they also exclude trees with unary nodes, and call the remaining trees reduced.

[1] It seems that the category they study is not the same as the category they define: their definition 1.1.4 does not seem to exclude contraction of external edges. I mention this curiosity as an illustration of the subtlety of formalising arguments with trees.

14.1 P-trees and free monads

eemonad-constr

14.1.1 Construction of free monads. Let $\mathrm{tr}'(\mathsf{P})$ denote the set of (isomorphism classes of) P-trees with a marked input leaf, i.e. the set of diagrams

$$\begin{tikzcd}
1 \arrow[d] & 0 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & 0 \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & 1 \arrow[d] \\
T^0 \arrow[d] & T^2 \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & T^1 \arrow[r] \arrow[d] & T^0 \arrow[d] \\
P^0 & P^2 \arrow[l] \arrow[r] & P^1 \arrow[r] & P^0
\end{tikzcd}$$

modulo isomorphism. (The cartesianness of the upper right-hand square says the edge is a leaf.) The set $\mathrm{tr}'(\mathsf{P})$ is naturally an object of $\boldsymbol{Set}/P^0$, the structure map $\mathrm{tr}'(\mathsf{P}) \to P^0$ returning the decoration of the marked leaf. There is also the natural projection to $\mathrm{tr}(\mathsf{P})$ given by forgetting the mark. We get altogether a polynomial functor

$$P^0 \leftarrow \mathrm{tr}'(\mathsf{P}) \to \mathrm{tr}(\mathsf{P}) \to P^0$$

which we denote by $\overline{\mathsf{P}}$. Its value on a set $X \to P^0$ is the set of P-trees with leaves decorated in X. More precisely, for a P-tree S, denote by L_{S} the set of leaves of S, then

$$\overline{\mathsf{P}}(X) = \left\{ (\mathsf{S}, f) \mid \mathsf{S} \in \mathrm{tr}(\mathsf{P}), \begin{tikzcd}[ampersand replacement=\&] L_{\mathsf{S}} \arrow[rr, "f"] \arrow[dr] \& \& X \arrow[dl] \\ \& P^0 \& \end{tikzcd} \right\}.$$

The polynomial functor $\overline{\mathsf{P}}$ is naturally a monad: the multiplication map $\overline{\mathsf{P}}\,\overline{\mathsf{P}}(X) \to \overline{\mathsf{P}}(X)$ sends a P-tree T with leaves decorated by other P-trees to the P-tree obtained by grafting the other P-trees onto the leaves of T. Note that the compatibility condition on the decorations states that the root edges of the decorating trees are precisely the leaves of the bottom tree, so the grafting makes sense. The unit for the multiplication is the map $P^0 \to \overline{\mathsf{P}}(P^0)$ sending an edge x to the trivial P-tree decorated by $x \in P^0$.

The construction $\mathsf{P} \mapsto \overline{\mathsf{P}}$ is clearly functorial. If $\alpha : \mathsf{Q} \to \mathsf{P}$ is a morphism of polynomial endofunctors, it is clear that $\overline{\alpha}^1 : \mathrm{tr}(\mathsf{Q}) \to \mathrm{tr}(\mathsf{P})$ sends trivial trees to trivial trees, and it is also clear it is compatible with grafting. Hence $\overline{\alpha}$ is a monad map.

freemonad **14.1.2 Proposition.** *Let* P *be a polynomial endofunctor. The monad* $\overline{\mathsf{P}}$ *given by*

$$P^0 \leftarrow \mathsf{tr}'(\mathsf{P}) \rightarrow \mathsf{tr}(\mathsf{P}) \rightarrow P^0$$

is the free monad on P.

Proof. Given $X \to P^0$, put $W_X := \overline{\mathsf{P}}(X)$, the set of P-trees with leaves decorated in X. In other words,

$$W_X = \overline{\mathsf{P}}(X) = \left\{ (\mathsf{S}, f) \mid \mathsf{S} \in \mathsf{tr}(\mathsf{P}), \begin{tikzcd} L_{\mathsf{S}} \arrow[rr, "f"] \arrow[dr] & & X \arrow[dl] \\ & P^0 & \end{tikzcd} \right\},$$

where L_{S} denotes the set of leaves of a tree S. It follows from the argument of Lemma 12.4.6 that W_X is a least fixpoint for the endofunctor $X + \mathsf{P}$, i.e. an initial object in $(X{+}\mathsf{P})$***-alg*** $\simeq X{\downarrow}P$***-alg***. Via the inclusion $\mathsf{P} \subset X + \mathsf{P}$ it also becomes a P-algebra. The construction $X \mapsto W_X$ is clearly functorial and defines a functor

$$\begin{array}{rcl} F : \boldsymbol{Set}/P^0 & \longrightarrow & \mathsf{P}\text{-}\boldsymbol{alg} \\ X & \longmapsto & W_X. \end{array}$$

To say that W_X is initial in $(X{+}\mathsf{P})$***-alg*** $\simeq X{\downarrow}P$***-alg*** is equivalent to saying that F is left adjoint to the forgetful functor $U : \mathsf{P}\text{-}\boldsymbol{alg} \to \boldsymbol{Set}/P^0$, and therefore (e.g. by Barr-Wells [13, Theorem 4, p.311]), the generated monad $X \mapsto W_X$ is the free monad on P. □

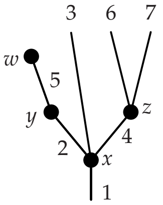
14.1.3 The free monad on a tree. We are particularly interested in the case where the polynomial endofunctor is itself a tree T. In this case we write $\mathsf{sub}(\mathsf{T})$ instead of $\mathsf{tr}(\mathsf{T})$, as we know that all maps between trees are injective. We restate this special case for emphasis:

14.1.4 Corollary. *Let* T *be a tree. The monad* $\overline{\mathsf{T}}$ *given by*

$$T^0 \leftarrow \mathsf{sub}'(\mathsf{T}) \rightarrow \mathsf{sub}(\mathsf{T}) \rightarrow T^0$$

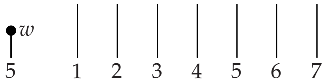
is the free monad on T.

14.1.5 Example. As an example of a computation of the free monad on a polynomial endofunctor, let us compute the free monad on the endofunctor associated to a tree. This is an easy example since it is a finite computation. So start with the polynomial endofunctor corresponding to some small tree, and compute the free monad. Let's take the tree



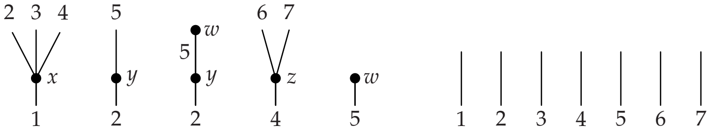
of example 12.2.13. WRITE OUT WHAT P IS

This is our first example with non-trivial types. As always, in the first round, where we compute the polynomial functor $\mathsf{Id} + P \circ 0$, we pick up all the nullary operations, and then add an unary operation for each type.



These correspond together to all the one-edge subtrees in the original tree.

In the next step of the iteration, we glue all these trees on top of all the nodes (the original operations), in all possible ways. In this way we get all the subtrees with edge-length 2, and again we add formally all the dotless trees, so altogether we get the subtrees of edge-length at most 2: Then the operations of the composite polynomial functor is the set of all ways of decorating the four bouquets



Next iteration we get also those with edge-length at most 3, which in this case is the set of all subtrees in the original tree.

In general, there will be only a finite number of steps in the iteration; after that it stabilises. This phenomenon is characteristic for trees: although the free-monad construction usually goes on infinitely, for trees it stops after step n where n is the height of the original tree.

Examples of polynomial monads from trees

14.1.6 Concrete description. Let us work out concretely what this polynomial functor does. The polynomial functor will have a variable X_i for each edge $i \in I$, and it will also have an output component P_j for each edge $j \in I$. The component P_j will be a sum indexed by all subtrees of T with root edge j.

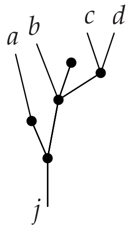
$$\big(X_i \mid i \in I\big) \mapsto \big(P_j(\mathbf{x}) \mid j \in I\big)$$

where each P_j is a sum over all subtrees with root j. For each such subtree with root j, the corresponding monomial is the product of all the variables corresponding to the leaves of that subtree. Given one such subtree S, say with set of leaves L, the corresponding monomial is the product

$$\prod_{l \in L} X_l,$$

the product of all the variables of the leaves of S.

If for example S is the subtree



then

$$P_j = X_a X_b X_c X_d.$$

All this is just the general description of what a polynomial functor does:

$$\big(X_i \mid i \in I\big) \mapsto \big(P_j(\mathbf{x}) \mid j \in I\big)$$

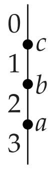
with

$$P_j = \sum_{b \in B_j} \prod_{e \in E_b} X_{s(e)}$$

Note that $X_{s(e)}$ is the variable corresponding to leaf e of $b \in B_j$, where $s(e)$ runs through the leaves of the subtree b (which has root edge j). So the description follows directly from the general description of polynomial functor.

14.1.7 Example. Many example, including the example *linear trees give linear polynomials*. We already saw in an example that the linear tree with n dots gives rise to the linear (directed) graph (with $n+1$ vertices and n edges. Now if we take the free monad on that one, that corresponds precisely to taking the free category on this graph it gives us precisely the category $[n]$ (the categorical n-simplex).

To be explicit, if the linear tree is



then the polynomial functor is the linear functor

$$\{0,1,2,3\} \leftarrow \{a,b,c\} \rightarrow \{0,1,2,3\}$$

there the two maps are input-edge and output-edge, respectively.

This corresponds to the directed graph $0 \xrightarrow{a} 1 \xrightarrow{b} 2 \xrightarrow{c} 3$.

Taking the free monad on this functor gives us a set of operations

$$\{00, 11, 22, 33, 01, 12, 23, 02, 13, 03\}$$

which are precisely the set of all arrows in the categorical 3-simplex [3], which is the free category on the previous graph.

14.2 The category *Tree*

Sec:Tree Tree

14.2.1 The category *Tree*. We define a larger category of trees ***Tree*** as the full subcategory of ***PolyMnd*** consisting of the free monads $\overline{\mathsf{T}}$, where T is a tree. This means taking the objects from ***TEmb*** and the morphisms from ***PolyMnd***. More precisely the category ***Tree*** is given by the Gabriel factorisation (identity-on-objects/fully-faithful) of ***TEmb*** $\rightarrow$ ***PolyMnd***:

$$\begin{tikzcd}
\textbf{\textit{Tree}} \arrow[r, "\text{f.f.}"] & \textbf{\textit{PolyMnd}} \arrow[d, shift left, "\text{forgetful}"] \arrow[d, phantom, "\dashv"] \\
\textbf{\textit{TEmb}} \arrow[u, "\text{i.o.}"] \arrow[r] & \textbf{\textit{PolyEnd}} \arrow[u, shift left, "\text{free}"]
\end{tikzcd} \tag{14.1}$$

Gab

The category ***Tree*** is equivalent to the category Ω introduced by Moerdijk and Weiss [83], whose presheaves are called dendroidal sets. The category ***Tree*** is also described as the Kleisli category of the free-forgetful adjunction restricted to ***TEmb***. The arrows in the category ***Tree*** are by definition monad maps $\overline{\mathsf{S}} \rightarrow \overline{\mathsf{T}}$. By adjunction these correspond to maps of endofunctors $\mathsf{S} \rightarrow \overline{\mathsf{T}}$, and many properties of the category ***Tree*** can be extracted in this viewpoint, without ever giving an explicit description of the monad maps. However, remarkably, the following result holds:

allmaps

14.2.2 Proposition. *All maps of endofunctors $\overline{\mathsf{S}} \rightarrow \overline{\mathsf{T}}$ are monad maps. In other words, the forgetful functor **Tree** $\rightarrow$ **PolyEnd** is full.*

This means that the maps in ***Tree*** have this surprisingly easy description: they are just commutative diagrams

$$\begin{tikzcd}
S^0 \arrow[d] & \mathrm{sub}'(\mathsf{S}) \arrow[l] \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathrm{sub}(\mathsf{S}) \arrow[r] \arrow[d] & S^0 \arrow[d] \\
T^0 & \mathrm{sub}'(\mathsf{T}) \arrow[l] \arrow[r] & \mathrm{sub}(\mathsf{T}) \arrow[r] & T^0.
\end{tikzcd} \tag{14.2}$$

mapinTree

Proof of the Proposition. Since the monad structure is defined in terms of unit trees and grafting, the assertion follows from the following two lemmas which are of independent interest.

□

`unittounit` **14.2.3 Lemma.** *Any map of polynomial endofunctors* $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ *maps trivial subtrees to trivial subtrees.*

Proof. If z is the root edge of a trivial subtree in S, then that trivial tree must map to a subtree of T with root $\phi(z)$, by commutativity of the right-hand square in (14.2). On the other hand, z is also the unique leaf of that trivial tree, and by commutativity of the left-hand square in (14.2), the unique leaf of the image tree must be $\phi(z)$. Hence the image tree has the property that its root is equal to its unique leaf, hence it is trivial. □

`graft-pres` **14.2.4 Proposition.** *Every morphism* $\phi : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$ *respects grafting. In other words, if a subtree* $\mathsf{R} \in \mathrm{sub}(\mathsf{S})$ *is a grafting* $\mathsf{R} = \mathsf{A} +_{\shortmid} \mathsf{B}$ *then the image subtree* $\phi^1(\mathsf{R}) \in \mathrm{sub}(\mathsf{T})$ *is given by* $\phi^1(\mathsf{R}) = \phi^1(\mathsf{A}) +_{\phi^1(\shortmid)} \phi^1(\mathsf{B})$.

Proof. The root of A is $\shortmid$ which is also a certain leaf of B. Hence the root of the image tree $\phi^1(\mathsf{A})$ is equal to $\phi^1(\shortmid)$ which is also a leaf of $\phi^1(\mathsf{B})$. Hence the grafting exists in T. It has root $\phi^0(\mathrm{root}(\mathsf{B}))$ as required, and set of leaves $\phi^0(\mathrm{leaves}(\mathsf{A}) + \mathrm{leaves}(\mathsf{B}) \smallsetminus \{\shortmid\})$. So it has the same boundary as the image of R, so by Lemma 12.3.17 they agree. □

`edge-map` **14.2.5 Lemma.** *A map of polynomial endofunctors* $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ *is completely determined by its value on the edge set.*

Proof. Let $\mathsf{R} \subset \mathsf{S}$ be an element of $\mathrm{sub}(\mathsf{S})$. The root of $\phi^1(\mathsf{R})$ must be the image of the root of R, by commutativity of the right-hand square of the representing diagram. Similarly, the set L_{R} of leaves of R is in bijection with the set of leaves of the image tree $\phi^1(\mathsf{R})$, by the cartesian condition on the middle square, but the latter set is also the image set $\phi(L_{\mathsf{R}})$, by commutativity of the left-hand square. Hence the set of leaves of $\phi^1(\mathsf{R})$ are fixed, so altogether the boundary of $\phi^1(\mathsf{R})$ is completely determined, and we conclude by Lemma 12.3.17. □

14.2.6 Corollary. *If* S *is nontrivial, every map* $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ *is determined by its value on one-node subtrees. More precisely, the map is the grafting of maps on those one-node trees, indexed by the inner edges of* S.

Proof. The first statement follows because the images of the nodes determine the images of their output and input edges, hence all edges have their image determined by the images of the nodes. For the more precise statement, note that the tree S is the grafting of its one-node trees indexed by its inner edges (cf. 12.3.24). The inner edges map to edges again, and since grafting is preserved, the whole map $\phi : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$ is the grafting of the restrictions of ϕ to the one-node subtrees (indexed by the inner edges of S). □

14.2.7 Proposition. *Let ϕ be an arrow in **Tree**. Then ϕ^0 preserves the tree order:*

$$x \leq y \quad \Rightarrow \quad \phi^0(x) \leq \phi^0(y).$$

Furthermore, if x and y are incomparable, then $\phi^0(x)$ and $\phi^0(y)$ are incomparable.

Proof. Suppose $x \leq y$. Let S denote the minimal subtree with y as root edge and x as a leaf. Having x as marked leaf makes S an element in $\mathrm{sub}'(\mathsf{T})$. By construction, $s(\mathsf{S}) = x$ and $t(p(\mathsf{S})) = y$. Now apply ϕ and use the fact that ϕ commutes with each of the structure maps. Hence $\phi^1(\mathsf{S})$ has $\phi^0(y)$ as root and $\phi^0(x)$ as marked leaf, and in particular $\phi^0(x) \leq \phi^0(y)$. For the second assertion, if x and y are incomparable, then by Lemma 12.3.12 there is a subtree in which x and y are leaves. Then $\phi^0(x)$ and $\phi^0(y)$ are leaves of the image subtree, and in particular incomparable. □

Note that ϕ^0 is not distance preserving, though, and that it is not necessarily injective. When it is injective it also reflects the tree order.

14.2.8 Remark. We should be more formal about the notion of minimal subtree spanned by two edges in ancestor relation, to avoid relying on intuition from geometric trees. Let the tree T be given by $A \leftarrow N' \rightarrow N \rightarrow A$. Suppose we have two edges in T in ancestor relation, so that $\rho^k(e) = r$ for some $k \in \mathbb{N}$. If $e = r$ is the root of T, then the minimal subtree is the one consisting only of the root. We aim at construction a subset $K \subset N$ with k elements, which intuitively is the set of dots in the path from e to r. Formally, for $0 \leq i < k$ the edge $\rho^i(e)$ is not the root, so we can consider it as an element in N'. The wanted set K is the set

$$K := \{p(\rho^i(e)) \mid 0 \leq i < k\}.$$

Now define $K' := N' \times_N K$ (this means we take all the input edges of all the dots we have), and finally let the set of edges be $K' + \{r\}$.

subtosub **14.2.9 Lemma.** *If $\phi : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$ is a map of trees, then $\phi^1 : \mathrm{sub}(\mathsf{S}) \to \mathrm{sub}(\mathsf{T})$ is inclusion preserving.*

Proof. The statement is that if $\mathsf{Q} \subset \mathsf{R}$ are elements in $\mathrm{sub}(\mathsf{S})$ then we have $\phi^1(\mathsf{Q}) \subset \phi^1(\mathsf{R})$ in $\mathrm{sub}(\mathsf{T})$. One way to see this is to observe that Q is determined by a subset of the nodes in R, cf. 12.3.8, and R is obtained from Q by grafting those complementary one-node trees onto Q. By preservation of grafting (14.2.4), $\phi^1(\mathsf{R})$ is therefore obtained from $\phi^1(\mathsf{Q})$ by grafting certain subtrees onto it, and in particular $\phi^1(\mathsf{Q}) \subset \phi^1(\mathsf{R})$. □

We have now gathered some basic knowledge of what general maps in ***Tree*** look like, and we already had a firm grip on the maps in ***TEmb***. The following proposition summarises various characterisations of the maps in ***TEmb*** from the viewpoint of ***Tree***, that is, it characterises the free maps:

free7 **14.2.10 Proposition.** *The following are equivalent for a map $\phi : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$.*

1. *ϕ is free (i.e. of the form $\overline{\alpha} : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$).*
2. *ϕ^0 is distance preserving.*
3. *The image of a one-node subtree is a one-node subtree.*
4. *For every subtree $\mathsf{R} \subset \mathsf{S}$, the image subtree $\phi^1(\mathsf{R}) \subset \mathsf{T}$ is isomorphic to R.*
5. *ϕ is injective, and if $\mathsf{R} \in \mathrm{sub}(\mathsf{T})$ is hit by ϕ^1 then all the subtrees of R are hit too.*
6. *ϕ is injective, and if $\mathsf{R} \in \mathrm{sub}(\mathsf{T})$ is hit by ϕ^1 then all edges of R are hit by ϕ^0.*
7. *ϕ is injective, and all edges in $\phi^1(\mathsf{S})$ are hit by ϕ^0.*

Proof. Straightforward verifications—omitted. □

14.2.11 Corollary. *In **Tree**, every isomorphism is free.* □

Another way to formulate Lemma 14.2.9 is that a map $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ restricts to any subtree $\mathsf{R} \subset \mathsf{S}$ to give a map $\overline{\mathsf{R}} \to \overline{\phi(\mathsf{R})}$. This is in fact a key observation, featured in the next proposition.

14.2.12 Boundary preserving maps. A map $\phi : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$ is called *boundary preserving* if it takes the maximal subtree to the maximal subtree. Equivalently, it takes leaves to leaves (bijectively) and root to root. It is clear that the composite of two boundary-preserving maps is boundary preserving, and that every isomorphism is boundary preserving.

surj **14.2.13 Lemma.** *Every surjection in **Tree** is boundary preserving.*

Proof. If $\phi^1 : \mathsf{sub}(\mathsf{S}) \to \mathsf{sub}(\mathsf{T})$ is surjective, in particular the maximal subtree $\mathsf{T} \in \mathsf{sub}(\mathsf{T})$ is hit, and since ϕ^1 is inclusion preserving by 14.2.9, $\mathsf{T} \in \mathsf{sub}(\mathsf{T})$ must be hit by $\mathsf{S} \in \mathsf{sub}(\mathsf{S})$. □

14.2.14 Proposition. *Every map of trees $\phi : \overline{\mathsf{S}} \to \overline{\mathsf{T}}$ factors essentially uniquely (i.e. uniquely up to unique isomorphism) as a boundary-preserving map followed by a free map. More precisely, the classes of boundary-preserving maps and free maps form an orthogonal factorisation system.*

We shall see in **??** that the boundary-preserving maps are precisely the *generic maps* in the sense of Weber [104] (defined in **??** below). Generic maps are characterised by a universal property. The proposition states that ***Tree*** has *generic factorisations*, an important property for a Kleisli category.

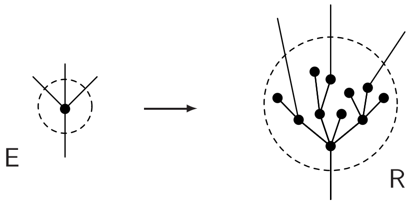
Proof. The first statement will be a special case of Proposition **??**, but here is the main argument: let $\mathsf{M} := \phi^1(\mathsf{S}) \in \mathsf{sub}(\mathsf{T})$ denote the image of the maximal subtree in S, and let $\alpha : \mathsf{M} \to \mathsf{T}$ be the inclusion—this is an map in ***TEmb***. Now $\overline{\alpha} : \overline{\mathsf{M}} \to \overline{\mathsf{T}}$ is the second factor in the wanted factorisation. Since ϕ^1 is inclusion preserving by Lemma 14.2.9, we get also a map $\overline{\mathsf{S}} \to \overline{\mathsf{M}}$ which is boundary preserving by construction. It is easy to see that this factorisation is unique (up to a unique isomorphism). Finally, since both classes of maps contain the isomorphisms and are closed under composition, we have an orthogonal factorisation system. □

14.2.15 Remark. There is a strong analogy between this boundary-preserving/free factorisation system in ***Tree*** and the root-preserving/ideal factorisation system in ***TEmb***: in both cases the left-hand component is characterised in terms of a certain max-preservation, while the right-hand component is characterised in terms of stability with respect to smaller elements, or equivalently in terms of preservation of certain minimal elements. Compare Lemma 12.3.10 with Proposition 14.2.10. We shall not pursue the analogy further.

We shall now describe the boundary-preserving maps in explicit terms. The main point of the analysis is Proposition 14.2.4 which says that any map out of a nontrivial tree is the grafting of its restriction to the one-node subtrees (also via Corollary 12.3.24). This is also just the content of the adjunction: to give a map $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ is equivalent to giving $\mathsf{S} \to \overline{\mathsf{T}}$, so we just have to say where each node goes.

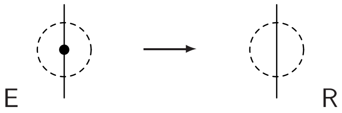
14.2.16 Boundary-preserving maps out of a one-node tree. Let E denote a one-node tree with n leaves, and suppose $\phi : \overline{\mathsf{E}} \to \overline{\mathsf{R}}$ is boundary preserving. By the cartesian condition, R necessarily has n leaves, and for any such R there are precisely $n!$ boundary-preserving maps from $\overline{\mathsf{E}}$ to $\overline{\mathsf{R}}$.

There are three cases to consider, depending on the number of nodes in R: If R has at least two nodes, then it has an inner edge, and since the map is boundary preserving this inner edge is not hit by ϕ^0, so ϕ is not surjective. Since in R the root is different from any leaf, ϕ^0 and hence ϕ is injective. Here is a picture of such a *node refinement*:



If R has precisely one node, clearly the map is an isomorphism. (This is not worth a picture.)

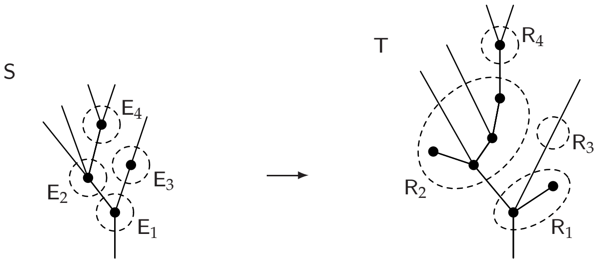
Finally there is a special case which occurs only for $n = 1$: then the tree R may be the trivial tree. In this case the two edges of E are both sent to the unique edge of R, and the node is sent to the maximal subtree (also trivial) by the boundary-preservation assumption. In this case, ϕ is clearly surjective (and not injective). Here is a picture of such a *unary-node deletion*:



14.2.17 Boundary-preserving maps, general case. Consider now a general boundary-preserving map $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$, and assume S is nontrivial. We know the map is the grafting of its restrictions to the one-node subtrees of S. Let E be a one-node subtree of S. We can factor the composite $\mathsf{E} \to \overline{\mathsf{S}} \to \overline{\mathsf{T}}$ into boundary-preserving followed by free:

$$\begin{tikzcd}
\overline{\mathsf{E}} \arrow[r] \arrow[d, "\text{b.pres.}"'] & \overline{\mathsf{S}} \arrow[d] \\
\overline{\mathsf{R}} \arrow[r, "\text{free}"'] & \overline{\mathsf{T}}.
\end{tikzcd}$$

The subtree $\mathsf{R} \in \mathrm{sub}(\mathsf{T})$ is the image of the subtree $\mathsf{E} \in \mathrm{sub}(\mathsf{S})$. The map $\overline{\mathsf{E}} \to \overline{\mathsf{R}}$ is either a node refinement or an unary-node deletion or an isomorphism. The original map $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ is the grafting of all the maps $\overline{\mathsf{E}} \to \overline{\mathsf{R}}$ for E running over the set of nodes in S. Here is a picture:



In conclusion, every boundary-preserving map $\overline{\mathsf{S}} \to \overline{\mathsf{T}}$ is the grafting of node refinements, unary-node deletions, and isomorphisms, indexed over the set of nodes in S. It is clear that we can realise the grafting by refining (or deleting) the nodes one by one in any order, and in particular we can first do all the unary-node deletions (this amounts to a surjection), then all the node refinements (this amounts to an injection).

Since surjections are boundary-preserving (14.2.13), and since node refinements are not surjective we find:

14.2.18 Lemma. *Every map in* ***Tree*** *factors essentially uniquely as a surjection followed by an injection. The surjections are generated by the unary-node deletions.*

□

Combining the two factorisation systems we get a double factorisation system:

14.2.19 Proposition. *Every morphism in* ***Tree*** *factors essentially uniquely as a surjection (a sequence of node deletions), followed by a boundary-preserving injection (a sequence of node refinements), followed by a free map (essentially a subtree inclusion).* □

14.2.20 Description of boundary-preserving injections into a given tree. To finish this first part of the paper, we show how to break the boundary-preserving injections into primitive maps. We already observed that we can refine one node at the time, but these are not the primitive maps.

In order to characterise the primitive boundary-preserving injections, we change the viewpoint, looking at maps into a given tree instead of out of it:

Fix a tree T, and suppose it has an inner edge $x = t(b) = s(e)$. We construct a new tree T/x by contracting the edge x, and exhibit a canonical boundary-preserving injection $\phi_x : \mathsf{T}/x \to \overline{\mathsf{T}}$:

$$\begin{tikzcd}
T^0 \smallsetminus \{x\} \arrow[d] & T^2 \smallsetminus \{x\} \arrow[l] \arrow[r] \arrow[d] & T^1/(b = p(e)) \arrow[r] \arrow[d] & T^0 \smallsetminus \{x\} \arrow[d] \\
T^0 & \mathrm{sub}'(\mathsf{T}) \arrow[l] \arrow[r] & \mathrm{sub}(\mathsf{T}) \arrow[r] & T^0
\end{tikzcd} \tag{14.3}$$

`contraction`

The maps are all obvious, except possibly $T^1/(b = p(e)) \to \mathrm{sub}(\mathsf{T})$: this map sends the node $b = p(e)$ to the two-node subtree containing b and $p(e)$, and sends all other nodes to their corresponding one-node subtree. It is clear that ϕ_X is boundary preserving and injective. The tree T/x has one inner edge less than T. We can now repeat the process for any subset of the inner edges in T, and for each subset we get a different boundary-preserving injection into T.

Conversely, every boundary-preserving injection $\mathsf{S} \to \mathsf{T}$ arises in this way. Indeed, we already know that these boundary-preserving injections are glued together from node refinements. The inner edges of the image trees form precisely the subset of edges we need to contract in order to recover the tree S.

In conclusion, we have derived explicit descriptions for each of the four classes of maps. The surjections can be described more explicitly as deletion of unary nodes, and each surjection can be broken into a composite of maps deleting just one node. The boundary-preserving injections are described as node refinements, and each boundary-preserving injection can be broken into a sequence of 'primitive' refinements adding just one new node. The free maps are the 'arity-preserving' tree embeddings, which also can be given add-one-node wise. The new node is added either at a leaf (in which case the map is root preserving), or at the root (in which case the map is an ideal embedding).

14.2.21 Linear trees. A *linear tree* is one in which every node has precisely one input edge. The full subcategory of ***Tree*** consisting of the linear trees is equivalent to the simplex category Δ. The factorisation systems restrict to Δ, recovering the well-known fact that every map in Δ factors uniquely as a surjection followed by a top-and-bottom-preserving injection, followed by distance-preserving injection. The primitive maps correspond to degeneracy and face maps in Δ, which motivates the terminology employed by Moerdijk and Weiss [83]. They call the unary-node deletions *degeneracy maps*. The primitive node refinements they call *inner face maps*, and the primitive tree embeddings *outer face maps*. The inner face maps play a crucial role in their theory, to express horn-filling conditions for dendroidal sets [82].

14.2.22 Remark. The factorisation of an injection into a generic injection followed by a free map has uniqueness. One can also always factor into free followed by generic injection, but this factorisation is not unique: if the generic injection ε is an expansion that takes place away from the subtree we first included (via ι), then that subtree will also be a subtree (with inclusion ι') in the expanded tree, so that

$$\varepsilon \circ \iota = \mathrm{id} \circ \iota'.$$

14.3 Trees of trees, constellations, and the Baez-Dolan construction

The following is mostly copied from [64].

To a given polynomial monad $P : \boldsymbol{Set}/I \to \boldsymbol{Set}/I$, represented by $I \leftarrow E \to B \to I$ we shall associate another polynomial monad $P^+ : \boldsymbol{Set}/B \to \boldsymbol{Set}/B$. The construction is originally due to Baez and Dolan [11], later given an elegant reformulation by Leinster [75]. The version here, for polynomial functors is from [64]. First we give a formal treatment, then an explicit graphical version, which also serves to establish that the output of the first version is polynomial.

Throughout this subsection, we fix a polynomial monad P, represented by

$$\begin{tikzcd} & E \arrow[r] \arrow[dl] & B \arrow[dr] & \\ I & & & I. \end{tikzcd}$$

BD-formal

14.3.1 The Baez-Dolan construction for polynomial monads, formal version. Denote by $\boldsymbol{PolyMnd}(I)$ the category of polynomial monads on $\boldsymbol{Set}/I$, i.e. the category of monoids in $\boldsymbol{Poly}^{\mathrm{c}}(I)$. Since P is a monad, the slice category $\boldsymbol{Poly}^{\mathrm{c}}(I)/P$ has a natural monoidal structure: the composite of $Q \to P$ with $R \to P$ is $R \circ Q \to P \circ P \to P$ and the unit is $\mathsf{Id} \to P$. Let $\boldsymbol{PolyMnd}(I)/P$ denote the category of polynomial monads over P, i.e. monoids in $\boldsymbol{Poly}^{\mathrm{c}}(I)/P$. The forgetful functor $\boldsymbol{PolyMnd}(I)/P \to \boldsymbol{Poly}^{\mathrm{c}}(I)/P$ has a left adjoint, the free P-monad functor, hence generating a monad $T : \boldsymbol{Poly}^{\mathrm{c}}(I)/P \to \boldsymbol{Poly}^{\mathrm{c}}(I)/P$. The Baez-Dolan construction consists in reinterpreting this as a monad on $\boldsymbol{Set}/B$ and observing that it is polynomial, cf. [64].

The key point is that there is a natural equivalence of categories

$$\boldsymbol{Poly}^{\mathrm{c}}(I)/P \xrightarrow{\sim} \boldsymbol{Set}/B, \tag{14.4}$$

given by evaluation at the terminal object $I \to I$, which we denote by 1. (See 10.3.)

Here we should really observe that at first we arrive at the slice category of $\boldsymbol{Set}/I$ over B/I, but then we reinterpret this as $\boldsymbol{Set}/B$.

In detail, if $Q \to P$ is an object in $\boldsymbol{Poly}^{\mathrm{c}}(I)/P$, the associated object in $\boldsymbol{Set}/B$ is simply $Q(1) \to P(1) = B$. The inverse equivalence basically takes an object $C \to B$ in $\boldsymbol{Set}/B$ to the object in $\boldsymbol{Poly}^{\mathrm{c}}(I)/P$ given by the fibre square

$$\begin{tikzcd}
 & E \times_B C \arrow[r] \arrow[d] & C \arrow[d] & \\
I & E \arrow[l] \arrow[r] & B \arrow[r] & I.
\end{tikzcd}$$

The promised monad, the Baez-Dolan construction on P, denoted $P^+ : \boldsymbol{Set}/B \to \boldsymbol{Set}/B$ is simply the monad corresponding to $T : \boldsymbol{Poly}^{\mathrm{c}}(I)/P \to \boldsymbol{Poly}^{\mathrm{c}}(I)/P$ under this equivalence. We shall describe this monad explicitly in a moment and see that it is polynomial.

To compare with Leinster's version of the Baez-Dolan construction [75], note that the above equivalence induces a monoidal structure on $\boldsymbol{Set}/B$ which is the *tensor product of P-collections*, for which the monoids are the *P-operads*, in the sense of [75]. The substitutional tensor product on $\boldsymbol{Set}/B$ is not the usual one (which we know corresponds to one-variable polynomial functors given just by $E \to B$) but rather some coloured version. Is it true at all that coloured operads are monoids in some monoidal category? YES, IN A BURRONI THING

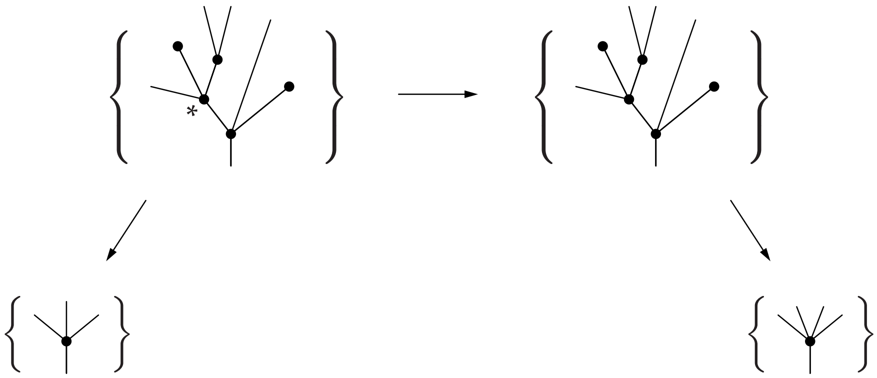
Hence we also get an equivalence of categories between ***PolyMnd***$(I)/P$ and the category of P-operads, and the free P-monad functor on ***Poly***$^{\mathrm{c}}(I)/P$ corresponds to the free P-operad functor used in Leinster's version of the Baez-Dolan construction.

MORE INFO:

BD **14.3.2 The Baez-Dolan construction for a polynomial monad, explicit graphical version.** Starting from our polynomial monad P, we describe explicitly a new polynomial monad P^+, shown afterwards to coincide with the one constructed above. The idea is to substitute into dots of trees instead of grafting at the leaves. Specifically, with B^* the set of P-trees, define U^* to be the set of P-trees with one marked node. There is now a polynomial functor

$$\begin{tikzcd} & U^* \arrow[r] \arrow[dl] & B^* \arrow[dr] & \\ B & & P^+ & B \end{tikzcd}$$

where $U^* \to B^*$ is the forgetful map, $U^* \to B$ returns the bouquet around the marked dot, and $t : B^* \to B$ comes from the monad structure on P—it amounts to contracting all dots back to the root dot (or setting a new dot in the dotless tree). Graphically:



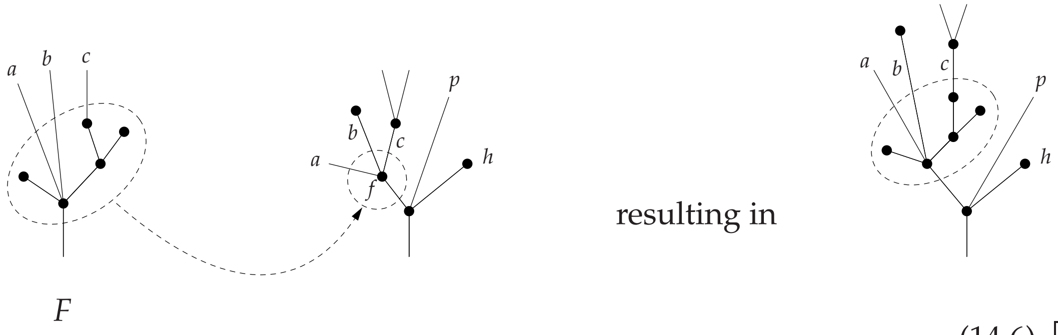
(14.5) slice1

(In this diagram as well as in the following diagrams of the same type, a symbol $\{\Psi\}$ is meant to designate the set of *all* bouquets like this (with the appropriate decoration), but at the same time the specific figures representing each set are chosen in such a way that they match under the structure maps.)

This polynomial endofunctor P^+ is naturally a monad: the substitution law can be described in terms of a partial composition law

$$B^* \times_B U^* \to B^*$$

defined by substituting a P-tree into the marked dot of an element in U^*, as indicated in this figure:



(14.6) BD-subst

Of course the substitution makes sense only if the decorations match. This means that $t(F)$, the 'total bouquet' of the tree F, is the same as the local bouquet of the node f. (The letters in the figure do not represent the decorations—they are rather unique labels to express the involved bijections.) The unit is given by the map $B \to B^*$ interpreting a bouquet as a tree with a single dot. (One can check again that this monad is cartesian.)

14.3.3 Comparison between the two versions of the construction. We wish to compare the two monads T and P^+ under the equivalence of categories $\boldsymbol{Poly}^{\mathrm{c}}(I)/P \overset{\sim}{\to} \boldsymbol{Set}/B$. The explicit description of P^+ allows us to compute its value on an object $C \to B$ of $\boldsymbol{Set}/B$: the result is the set of P-trees with each node decorated by an element of C, compatibly with the arity map $C \to B$ (being a P-tree means in particular that each node already has a B-decoration; these decorations must match). We claim that this is the same thing as a Q-tree, where Q is the polynomial functor corresponding to $C \to B$ under the key equivalence, i.e. given by

$I \leftarrow E \times_B C \to C \to I$. Indeed, since the tree is already a P-tree, we already have I-decorations on edges, as well as bijections for each node between the input edges and the fibre E_b over the decorating element $b \in B$. But if $c \in C$ decorates this same node, then the cartesian square specifies a bijection between the fibre over c and the fibre E_b and hence also with the set of input edges. So in conclusion, P^+ sends C to the set of C-trees. On the other hand, T sends the corresponding polynomial functor Q to the free monad on Q, with structure map to P given by the monad structure on P. Specifically, T produces from Q the polynomial monad

$$\begin{tikzcd}
{} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & C* \arrow[d] \\
E* \arrow[r] \arrow[d] & B* \arrow[d] \\
E \arrow[r] & B
\end{tikzcd}$$

where $C*$ denotes the set of Q-trees, so the two endofunctors agree on objects. The same argument works for arrows, so the two endofunctors agree.

To see the monad structures agree, note that the set of operations for $P^+ \circ P^+$ is the set of P-trees with nodes decorated by P-trees in such a way that the total bouquet of the decorating tree matches the local bouquet of the node it decorates. The composition law $P^+ \circ P^+ \Rightarrow P^+$ consists in substituting each tree into the node it decorates. On the other hand, to describe the monad T it is enough to look at the base sets, since each top set is determined as fibre product with E over B. In this optic, T sends B to $B*$, and $T \circ T$ sends B to $B**$ which is the set of $P*$-trees, which is the same as P-trees with nodes decorated by P-trees, and edges decorated in I, subject to the usual compatibility conditions. Clearly the composition law $T \circ T \Rightarrow T$ corresponds precisely to the one we described for P^+. For both monads, the unit is described as associating to a bouquet the corresponding one-dot tree.

In conclusion, the two constructions agree, and, in particular, produce a polynomial functor.

MORE STUFF FROM ZOOM

Part III

Categorical polynomial functors

14.4 Introduction

As we have mentioned at several occasions in Part II, the notion of polynomial functor makes sense quite generally in any locally cartesian closed category $\mathscr{E}$. However, one context where it would be really nice to have polynomial functors is in ***Cat***, and ***Cat*** is not locally cartesian closed (cf. analysis below). Nevertheless there are several useful ways of making sense of the notion of polynomial functors in ***Cat***.

We are also quite interested in the case of polynomial functors in the category of groupoids (not locally cartesian closed either), because with groupoids we can fix some of the problems we had with sets, related to automorphisms. With groupoids we will be able to 'represent' structures we could not deal with effectively in ***Set***. In example 15.3.1 below we study the family functor which helps making the notion of symmetric operad representable, a thing we could not handle in ***Set***. We will also be able to incorporate species into the picture.

There are at least two versions of this notion of polynomial functor for ***Cat***. One is the slice viewpoint (Chapter 15) where diagrams of categories and functors like

$$\begin{tikzcd} I & E \arrow[l, "s"'] \arrow[r, "p"] & B \arrow[r, "t"] & J \end{tikzcd} \tag{14.7}$$

shape

define functors between slices like this:

$$\begin{tikzcd} \mathbf{Cat}/I \arrow[r, "s_*"] & \mathbf{Cat}/E \arrow[r, "p_*"] & \mathbf{Cat}/B \arrow[r, "t_!"] & \mathbf{Cat}/J \end{tikzcd}$$

Here a special condition is required on the functor $p : E \to B$ in order for the pushforth p_* to exist. This is the Conduché condition which we study in detail in 15.1.

The other viewpoint is about presheaves (Chapter **??**), where a diagram (14.7) defines a functor between presheaf categories

$$\begin{tikzcd} \mathbf{PrSh}(I) \arrow[r, "s^*"] & \mathbf{PrSh}(E) \arrow[r, "p_*"] & \mathbf{PrSh}(B) \arrow[r, "t_!"] & \mathbf{PrSh}(J) \end{tikzcd}$$

In this case, the adjoint functors are different, but they still have the same formal properties. The presheaf category setting is closely related to a special variant of the slice viewpoint: a presheaf on a category I is the same thing as a discrete fibration $F \to I$, which is a special case of an object in ***Cat***$/I$. However, the functors do not quite agree in general...

There is an intermediate possibility which is to look at groupoid-valued presheaves, or rather prestacks...

In any case this second viewpoint relies on the notions of left Kan extension, final functors (and the factorisation system consisting of final functors and discrete fibrations). There are two formulations of this theory: in terms of presheaves and in terms of discrete fibrations. For presheaves the adjunctions are straightforward to use. In the discrete-fibration viewpoint the pushforth functor is not the usual one because the pushforth of a discrete fibration is not in general a discrete fibration again. You need to factor it into final followed by discrete. . . This slightly more complicated pushforth corresponds to the usual pushforth of presheaves.

Chapter 15

[Polynomial functors for slices of *Cat*]

Ch:Cat/I

Cat is not locally cartesian closed

Another way of formulating the problem with ***Cat*** is to say that there are functors $p : E \to B$ for which the pullback functor

$$\begin{aligned} p_* : \textit{\textbf{Cat}}/B &\longrightarrow \textit{\textbf{Cat}}/E \\ X &\longmapsto E \times_B X \end{aligned}$$

does not have a right adjoint. Clearly this is a serious problem, since the whole construction of polynomial functors is based on such right adjoints.

Here is an example which in a precise sense is the initial example—the example that has precisely what it takes to exhibit the problem, and nothing more than that.

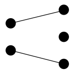
It is not difficult to see that a pullback functor has a right adjoint if and only if it preserves all colimits. Indeed, any functor that has a right adjoint preserves all colimits, and the converse is true if just since ***Cat*** is locally presentable.

First we'll need some simplicial notions and notations.

15.0.1 Notation. Let $[n]$ denote the category freely generated by a string of n arrows. It is the category corresponding to the totally ordered set $[n] = \{0 < 1 < \cdots < n\}$. The full subcategory of ***Cat*** consisting of these categories is Δ.

triangle

15.0.2 Example. This is an example of a functor $p : E \to B$ such that p^* does not preserve colimits, and in particular p^* cannot have a right adjoint. Let E be the category $[1]$, and let B be $[2]$, and let the functor $p : E \to B$ be the functor $[1] \to [2]$ which preserves the endpoints:



$[1] \to [2]$

So this is the inclusion of the hypotenuse $0 \to 2$ in the triangle.

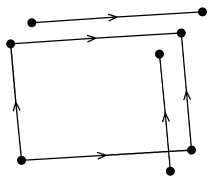
$$\begin{tikzcd}
0 \arrow[rr] \arrow[dr] & & 2 \arrow[dl] \\
& 1 &
\end{tikzcd}$$

We claim that the corresponding pullback functor,

$$p^* : \boldsymbol{Cat}/B \to \boldsymbol{Cat}/E$$

does not preserve colimits: we exhibit a concrete pushout square which is not preserved:

$$\begin{tikzcd}
{[1]} \arrow[r] & {[2]} \\
{[0]} \arrow[r] \arrow[u] \arrow[ur, phantom, "\lrcorner", very near end] & {[1]} \arrow[u]
\end{tikzcd}$$



(15.1) Segal-square

These four maps are maps in $\boldsymbol{Cat}/B$. Now pull back this square to $\boldsymbol{Cat}/E$. We get

$$\begin{tikzcd}
{[0]} \arrow[r] & {[1]} \\
\emptyset \arrow[r] \arrow[u] & {[0]} \arrow[u]
\end{tikzcd}$$

which is clearly not a pushout, because the pushout of • with • over the empty set is just the disjoint union of two points, whereas in $[1]$ there is an arrow between the two points.

15.1 Conduché fibrations

Sec:Conduche

Question: when does the pullback functor $p^* : \boldsymbol{Cat}/B \to \boldsymbol{Cat}/E$ have a right adjoint? The problem in the example we just saw is that there is a factorisation in the bottom space that cannot be lifted to the top space. The Conduché condition will preclude this sort of problem:

15.1.1 The Conduché condition. Let $p : E \to B$ be a functor. The *Conduché condition* states that for every triangle $h = g \circ f$ in B, every lifting of the hypotenuse h to E extends to a lift of the whole triangle. Furthermore, this lift is unique in the sense that if there are two such lifts, then there is a chain of vertical arrows comparing them. In other words, if triangle $h' = g' \circ f'$ and $h' = g'' \circ f''$ both lie over $h = g \circ f$, then there is a vertical arrow i such that $f'' = i \circ f'$ and $g'' = i \circ g'$, or possibly i is not a single vertical arrow but a zigzag of vertical arrows

$$\begin{tikzcd}
\cdot \arrow[rr, "h'"] \arrow[dr, "f'"'] \arrow[ddr, "f''"'] & & \cdot \\
& \cdot \arrow[ur, "g'"'] \arrow[d, dash, dotted] & \\
& \cdot \arrow[uur, "g''"'] &
\end{tikzcd}$$

Clearly, Example 15.0.2 above does not satisfy the Conduché condition.

Examples of functors that satisfy Conduché condition are fibrations and opfibrations.

15.1.2 Example. A Grothendieck fibration $p : E \to B$ is a Conduché fibration. Indeed, given a triangle $\gamma = \psi \circ \varphi$ in B and a lift to E of the hypotenuse γ. We need to lift the rest of the triangle. Take the cartesian lift of ψ to the target of the lift of γ. By the cartesian property, there is then a unique lift of φ to complete the triangle above. So every factorisation lifts. To see uniqueness, if we are given any other lift of the triangle then the cartesian property of the first one we have ensures that there is a comparison. So between any two lifts of the triangle there is a comparison to the cartesian lift, and hence a zigzag of comparisons.

15.1.3 Theorem. *For a functor $p : E \to B$ the following are equivalent.*

(i) The pullback functor $p^* : \mathbf{Cat}/B \to \mathbf{Cat}/E$ has a right adjoint.
(ii) p^* preserves colimits (i.e. p^* is cocontinuous).
(iii) p satisfies the Conduché condition.

This theorem is due to Giraud, who gave a very detailed proof in the beginning of the sixties. Unfortunately the result was buried deep in the long and very dry memoir *Méthode de la descente* [44]. Ten years later the result was rediscovered by Conduché [29] (unaware that the result was already in [44]), and today the condition bears his name.

15.1.4 Remark. To say that p^* has a right adjoint is equivalent to saying that the functor $J \mapsto \mathrm{Hom}_E(p^* J, X)$ is representable for every X. To prove that a functor $\mathbf{Cat}^{\mathrm{op}} \to \mathbf{Set}$ is representable, there are some tricks, extracted from Giraud [44]. Since ***Cat*** is locally presentable, it is enough to prove that the functor sends colimits to limits, and some of them are automatic in our case (I think that arbitrary coproducts are preserved, and also coequalisers along a regular epi, or something of the sort, see [44] Section 2.) Under these conditions, it only remains to see that pushouts like Diagram (15.1) are sent to pullbacks. Perhaps a theorem like this can be extracted: for a functor $F : \mathbf{Cat}^{\mathrm{op}} \to \mathbf{Set}$ sending certain colimits to limits, you can check if it preserves all colimits by restricting to the truncated $\Delta_2 = \{[0], [1], [2]\}$ and see if this truncated simplicial set is the nerve of a category. The Segal condition is more or less equivalent to the Conduché condition. . . ? NOT SURE IF ANYTHING IS GOING TO COME OUT OF THIS PARAGRAPH

15.1.5 The Conduché condition in simplicial terms. Let us see what the Conduché condition has to do with the issue. First let us describe some subcategories in E. We are interested in what happens above this triangular subcategory of B

$$\begin{tikzcd}
b_0 \arrow[rr, "\gamma"] \arrow[dr, "\varphi"'] & & b_2 \\
& b_1 \arrow[ur, "\psi"'] \arrow[u, phantom, "t"] &
\end{tikzcd}$$

We see this subcategory $t \subset B$ as the image of a functor $[2] \to B$, and we will use the pushout description of the category $[2]$ given in Diagram (15.1) in Example 15.0.2. This means that b_1 is the image of $[0]$, and that the arrows φ and ψ are images of the two copies of $[1]$.

We consider the three fibre categories E_{b_0}, E_{b_1}, and E_{b_2}. For each of the three arrows we also consider the category of arrows lying over the arrow (to be described in more detail in a moment), let us describe the subcategory $E_\varphi \subset E$ of arrows lying over some $\varphi : b_0 \to b_1$. The objects of E_φ are the union $E_{b_0} \coprod E_{b_1}$. The arrows are those that either lie over φ or lie over the identity arrows of b_0 and b_1. A more concise description goes as follows. Note first that the set of arrows (or the category of arrows) in B can be described as the set (or category) of functors $I \to B$, where $I = \Delta[1]$ is the interval category with two objects and a single non-identity arrow. For an arrow $\varphi : b_0 \to b_1$ in B we will use the same letter φ to denote the corresponding map $I \to B$. Now E_φ is simply the pullback

$$\begin{tikzcd}
E_\varphi \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d] \\
I \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

Obviously E_φ contains E_{b_0} and E_{b_1} as full subcategories. In simplicial terms, this is just the pullback square (here shown for b_0):

$$\begin{tikzcd}
E_{b_0} \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E_\varphi \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d] \\
\Delta[0] \arrow[r, "0"'] & \Delta[1] \arrow[r, "\varphi"'] & B
\end{tikzcd}$$

Finally we consider the category E_t whose arrows are those lying over some arrow in the triangle.

Each of these categories can be described as a pullback

$$\begin{tikzcd}
E_i \arrow[r] \arrow[d] & E \arrow[d] \\
{[i]} \arrow[r] & B
\end{tikzcd}$$

for $i = 0, 1, 2$. For $i = 0$ it just says that E_0 is the fibre over the object singled out by $[0] \to B$. For an arrow, . . . and finally for a triangle.

So here is a picture of the seven categories. All the arrows between them are full inclusions:

$$\begin{tikzcd}
E_{b_0} \arrow[rr] \arrow[ddr] & & E_\gamma \arrow[d] & & E_{b_2} \arrow[ll] \arrow[ddl] \\
 & & E_t & & \\
 & E_\varphi \arrow[ur] & & E_\psi \arrow[ul] & \\
 & & E_{b_1} \arrow[ul] \arrow[ur] & &
\end{tikzcd}$$

15.1.6 Main lemma. *The Conduché condition is equivalent to saying that the bottom square is a pushout.*

Suppose the right adjoint p_* exists. Then p^* preserves all colimits. In particular it preserves the square (15.1) mapping into B singling out the triangle t. So we get a pushout diagram

$$\begin{tikzcd}
p^*[1] \arrow[r] & p^*[2] \\
p^*[0] \arrow[r] \arrow[u] & p^*[1] \arrow[u]
\end{tikzcd}$$

In other words,

$$\begin{tikzcd}
E_\varphi \arrow[r] & E_t \\
E_{b_1} \arrow[r] \arrow[u] & E_\psi \arrow[u]
\end{tikzcd}$$

CLAIM: *to say that this is a pushout is equivalent to saying that the natural map of distributors*

$$E_\varphi \otimes_{E_{b_1}} E_\psi \to E_\gamma$$

is an isomorphism, where γ is the composite $\psi \circ \varphi$.

This in turn is to say that the Conduché condition holds!

So concerning the claim: by the universal property of the pushout, there is always a map $E_\varphi \coprod_{E_{b_1}} E_\psi \to E_t$. To see that this is an equivalence, it is enough to look at the various hom sets. Those over φ and ψ are obvious. It remains to see what happens over the long edge:

$$(E_\varphi \coprod_{E_{b_1}} E_\psi)_{|\gamma} \to (E_t)_{|\gamma} = E_\gamma.$$

So the next lemma explains the claim.

15.1.7 Main lemma. *The restriction of the pushout* $(E_\varphi \coprod_{E_{b_1}} E_\psi)_{|\gamma}$ *is precisely the tensor product of distributors* $E_\varphi \otimes_{E_{b_1}} E_\psi$.

Proof. Just compute: both the pushout and the tensor product are computed as a coequaliser, where you divide out by the action of those vertical maps in E_{b_1} in the middle. . . □

t-construction **15.1.8 Concrete description of** p_*X**.** Here is the description of the category p_*X, which is a category over B. The objects over $b \in B$ are diagrams

$$\begin{tikzcd}
& X \arrow[d] \\
E_b \arrow[ur, "s"] \arrow[r, phantom, "\subset"] & E
\end{tikzcd}$$

(i.e., E-arrows $E_b \to X$). So an object is a pair (b, s), where $b \in B$ and where s is as described.

The arrows are diagrams

$$\begin{tikzcd}
& X \arrow[d] \\
E_\varphi \arrow[ur, "\Phi"] \arrow[r, phantom, "\subset"] & E
\end{tikzcd}$$

The source of such a Φ are given by restricting to $E_{b_0} \subset E_\varphi$, and the target is given by restricting to $E_{b_1} \subset E_\varphi$.

So we can describe the arrows of $p_* X$ in this more general manner (without favouring a particular b): the arrows are diagrams

$$\begin{tikzcd}
 & X \arrow[d] \\
{[1] \times_{[0]} E} \arrow[ur, "s"] \arrow[r] \arrow[d] & E \arrow[d] \\
{[1]} \arrow[r] & B
\end{tikzcd}$$

Let us describe composition of arrows in $p_* X$. Given two composable arrows (φ, Φ) and (ψ, Ψ) lying over our fixed composable pair $\psi \circ \varphi = \gamma$ in B. Then we have a diagram

$$\begin{tikzcd}
E_\varphi \arrow[r] & E_t & \simeq E_\varphi \sqcup_{E_{b_1}} E_\psi \\
E_{b_1} \arrow[r] \arrow[u] & E_\psi \arrow[u] &
\end{tikzcd}$$

It is the Conduché condition that guarantees the isomorphism $E_t \simeq E_\varphi \sqcup_{E_{b_1}} E_\psi$. Note that E_γ sits inside E_t...

Now by the universal property of the pushout, our maps $\Phi : E_\varphi \to X$ and $\Psi : E_\psi \to X$ glue to a map on $E_\varphi \coprod_{E_{x_1}} E_\psi = E_t$ (still a E-map). Finally restrict this map $E_t \to X$ to $E_\gamma \subset E_t$.

Checking associativity of this composition law is straightforward and unenlightening, and describing the identity arrows is also trivial.

15.1.9 In terms of distributors. A category D with a functor to $I = [1]$ can be interpreted as a distributor. Namely, it contains two full subcategories D_0 and D_1. (The category D_0 is a sieve in D, which means that if an arrow $\xrightarrow{f}$ is in D_0 then any composition $\to\xrightarrow{f}$ is in D_0 too.) There may be arrows from an object in D_0 to an object in D_1, but no arrows in the other direction.

These arrows express a correspondence, which is equivalent to giving a distributor $D_0^{\mathrm{op}} \times D_1 \to \textbf{\textit{Set}}$. We also call this distributor D.

Recall what is the tensor product of distributors: given $D : A^{\mathrm{op}} \times B \to \textbf{\textit{Set}}$ and $E : B^{\mathrm{op}} \times C \to \textbf{\textit{Set}}$, then we can form $D \otimes_B E$. It is the set of pairs where f goes from A to B, and g goes from $B \to C$, and we identify $f \otimes ug$ with $fu \otimes g$.

The Conduché condition states precisely that for every triangle $\gamma = \psi \circ \varphi$ in B, the natural map $E_\varphi \otimes_{E_{b_1}} E_\psi \to E_\gamma$ is an isomorphism.

If a category X has a functor to $[2]$, then there are defined three distributors D_{01}, D_{12}, and D_{02}, which are the pullback of X along the face maps inclusions $[1] \hookrightarrow [2]$. And then there is a composition law $D_{01} \otimes_{D_1} D_{12} \to D_{02}$. (Under the conditions or always?) To say that the square image of 15.1 is a pushout is equivalent to saying that this natural map is an isomorphism.

15.1.10 More advanced proof of the Theorem. This proof identifies the Conduché maps as generators for the localisation $\boldsymbol{sSet} \to \boldsymbol{Cat}$.

We fix a functor $p : E \to B$. We want to find the conditions under which p^* has a right adjoint.

Let $N : \boldsymbol{Cat} \to \boldsymbol{sSet}$ be the nerve functor. Consider the square

$$\begin{tikzcd}
\boldsymbol{Cat}/E \arrow[d, "N"'] & \boldsymbol{Cat}/B \arrow[l, "p^*"'] \arrow[d, "N"] \\
\boldsymbol{sSet}/NE & \boldsymbol{sSet}/NB \arrow[l, "(Np)^*"]
\end{tikzcd}$$

Now the bottom map always has a right adjoint because we are talking slices of a presheaf category. Now we cannot use N to compare this to fact to a statement about p^* because N is not cocontinuous. Instead we shall use the left adjoint to N, the fundamental category functor

$$\tau_1 : \boldsymbol{sSet} \to \boldsymbol{Cat}$$

Since this is a left adjoint it is cocontinuous. So now the relevant diagram is

$$\begin{tikzcd}
\boldsymbol{Cat}/E & \boldsymbol{Cat}/B \arrow[l, "p^*"'] \\
\boldsymbol{sSet}/NE \arrow[u, "\tau_1"] & \boldsymbol{sSet}/NB \arrow[l, "(Np)^*"] \arrow[u, "\tau_1"']
\end{tikzcd} \tag{15.2}$$

tau-square

Definition. An arrow in $\boldsymbol{sSet}$ is called a quasi-isomorphism if its image in $\boldsymbol{Cat}$ under τ_1 is an isomorphism.

Now we shall invoke some general theory about localisation of cocomplete categories (note that both ***Cat***, ***sSet***, and their slices are cocomplete).

Let $F : \mathscr{C} \to \mathscr{D}$ be a cocontinuous functor between cocomplete categories. Let Σ denote the class of arrows in $\mathscr{C}$ which are sent to isomorphisms in $\mathscr{D}$. Then there is a cocomplete category $\Sigma^{-1}\mathscr{C}$ with a cocontinuous functor $\mathscr{C} \to \Sigma^{-1}\mathscr{C}$ such that for every functor that inverts Σ factors uniquely through $\Sigma^{-1}\mathscr{C}$.

Recall that for an arrow $f : A \to B$, an object X is (strictly) orthogonal to f, written $f \perp X$, if

$$\begin{tikzcd} A \arrow[r] \arrow[d, "f"'] & X \\ B \arrow[ur, dashed, "\exists!"'] & \end{tikzcd}$$

Now let $\mathscr{C}' \subset \mathscr{C}$ denote the full subcategory whose objects are the X orthogonal to all $f \in \Sigma$.

15.1.11 Theorem. *(See Borceux, Thm 5.4.8 or something EXACT REFERENCE?) The category $\mathscr{C}'$ is a reflexive subcategory in $\mathscr{C}$, and the reflection $r : \mathscr{C} \to \mathscr{C}'$ is the localisation with respect to Σ.*

That is, for every $\mathscr{C} \to \mathscr{D}$ (cocontinuous between cocomplete categories) which inverts the arrows in Σ, there is a factorisation

$$\begin{tikzcd} \mathscr{C} \arrow[rr] \arrow[dr] & & \mathscr{D} \\ & \mathscr{C}' \arrow[ur, dashed, "\exists!"'] & \end{tikzcd}$$

So much for the general theory. In our case, $\tau_1 : \textbf{\textit{sSet}} \to \textbf{\textit{Cat}}$ is the localisation and the reflector is N. Sigma is the set of map

$$I^k \to [k]$$

where the simplicial set I^k is the graph consisting of k arrows in a sequence (but without their composites), while $[k]$ is the simplicial set $\Delta[k]$, the k-simplex, which is a category. Another way of describing I^k is as the coproduct in ***sSet*** of intervals. The maps $I^k \subset [k]$ describe the principal edges of a simplex.

Claim: $(Np)^*$ preserves quasi-isomorphisms provided the Conduché condition holds.

If this is the case, both ways around in the diagram (15.2) invert quasi-isomorphisms.

Now consider the maps

$$\begin{tikzcd}
(Np)^*X \arrow[rr] \arrow[dr] & & N(p^*\tau_1 X) \arrow[dl] & & X \arrow[rr, "\varepsilon"] \arrow[dr] & & N\tau_1 X \arrow[dl] \\
& NE \arrow[rrrr, "Np"'] & & & & B &
\end{tikzcd}$$

The map ε is a quasi-isomorphism (check this!). Hence the map $(Np)^*X \longrightarrow N(p^*\tau_1 X)$ is an isomorphism. This shows that at least the Diagram (15.2) commutes up to isomorphism.

It remains to show that p^* preserves colimits. Consider some colimit $(S_i)_{i\in I} \to S$ in $\boldsymbol{Cat}/B$, and consider also the colimit $(NS_i)_{i\in I} \to T$ in $\boldsymbol{sSet}/NB$. Note that this is not the image of the first under N, because N does not preserve colimits. However, taking τ_1 on this second colimit reproduces the first. So going the two ways around in Diagram (15.2) we get some diagram which is a colimit by going the lower way around, and which is p^* of the original colimit the upper way around. Hence p^* preserves the colimit. End of proof.

15.1.12 Remark. One could ask other questions: for example given any functor $p : E \to B$ ask whether the pseudo-pullback has a right adjoint. And the notion of right adjoint might be weakened as follows: instead of requiring a bijection of hom sets,

$$\mathrm{Hom}_E(p^*Y, X) \simeq \mathrm{Hom}_B(Y, p_*X),$$

see if we cannot define a condition where instead some hom cats should be equivalent . . .

15.2 Polynomial functors in *Cat*

Definition. A polynomial functor $\boldsymbol{Cat}/I \to \boldsymbol{Cat}/J$ is one of the form

$$\boldsymbol{Cat}/I \xrightarrow{s^*} \boldsymbol{Cat}/E \xrightarrow{p_*} \boldsymbol{Cat}/B \xrightarrow{t_!} \boldsymbol{Cat}/J$$

for a diagram

$$\begin{tikzcd}
& E \arrow[r, "p"] \arrow[dl, "s"'] & B \arrow[dr, "t"] & \\
I & & & J
\end{tikzcd}$$

where p is a Conduché fibration.

A polynomial functor in one variable is just the special case where I and J are the terminal category. In other words, it is a functor of the form

$$\begin{array}{rcl} \boldsymbol{Cat} & \longrightarrow & \boldsymbol{Cat} \\ C & \longmapsto & p_*(C \times_B E) \end{array}$$

15.2.1 Beck-Chevalley conditions. The Beck-Chevalley conditions hold for the adjunction $p_! \dashv p^*$, and consequently it holds also for the adjunction $p^* \dashv p_*$ whenever this last functor exists.

The Beck-Chevalley condition ensures that most things can be checked fibrewise. Some squares to draw:

$$\begin{tikzcd}
\boldsymbol{Cat}/E_b \arrow[r] & \boldsymbol{Cat}/E \\
\boldsymbol{Cat} \arrow[u] \arrow[r] \arrow[ur, phantom, very near start, "\urcorner"] & \boldsymbol{Cat}/B \arrow[u]
\end{tikzcd}$$

$$\begin{tikzcd}
E_b \arrow[r] \arrow[d] \arrow[dr, phantom, very near start, "\lrcorner"] & E \arrow[d, "p"] \\
1 \arrow[r, "\ulcorner b \urcorner"'] & B
\end{tikzcd}$$

15.2.2 Lemma. *The pullback of a Conduché fibration is again a Conduché fibration.*

The trickier point is whether there is a distributivity law. But this should be true.

15.2.3 Proposition. *The distributive law holds in the same form as in 8.3.4.*

The fibrewise argument allows us to reduce to the case $C = 1$ again. Then $B \to 1$ is a discrete category. Then all the fibre arguments should work again.

15.2.4 More abstract approach. In fact all this may work in a more abstract setting. Let $\mathscr{C}$ be a category with pullbacks. Define a notion of proper map: a map φ is *proper* if pullback along it has a right adjoint and if the same if true for every pullback of φ. Then it should be possible to deduce Beck-Chevalley and distributivity.

The terminology 'proper map' is perhaps not necessary. Perhaps we should just use the word *exponentiable*.

[Note that in the treatment of representable natural transformations (10.1.8), we used lowerstarring along some map w which appeared in between two exponentiable maps. So we would like to have the property that is $q = p \circ w$, and if p and q are exponentiable, then also w is. . . This property holds for discrete fibrations, for example, or any class that is the right-hand class of a factorisation system.]

One possible way to argue is this: consider the Yoneda embedding $y : \mathscr{C} \to \widehat{\mathscr{C}}$. In $\widehat{\mathscr{C}}$ everything works. If we can just show that p_* is sent to $_*$. (in analogy with the fact that y preserves exponentiation: it preserves all those exponentiation that exist. In the same way, it should be true that y preserves all those lowerstars that happen to exist. . .

one reason this works is that the sum map is just composition. In other contexts it might be something more complicated, and then it might not work. Suppose for example we work in the category of discrete fibrations. Then the left adjoint might involve some factorisation (every functor factors as a final map followed by a discrete fibration), and the lowershriek would be this last part only. In such cases where the sum map is not pure composition we might get trouble Another example: sheaves: lowershriek of a sheaf is not just composition: we need to sheafify afterwards!

15.3 The family functor

Bij **15.3.1 Finite sets and bijections.** Let $\mathbb{B}$ denote the category of finite sets and bijections, let $\mathbb{E}$ denote the category of pointed finite sets and bijections preserving the basepoint, and denote the forgetful functor

$$p : \mathbb{E} \to \mathbb{B}.$$

Given an arrow $\varphi : I \to J$ (i.e., a bijection) and a marked point e in J (i.e., a point in the fibre $\mathbb{E}_J$, then there is a unique way of picking a marked point in I such that φ becomes a mark-preserving map—indeed since φ is a bijection just take the inverse image of e. This shows that arrows in $\mathbb{B}$ have unique lifts to $\mathbb{E}$, which is one characterisation of being a discrete fibration, and hence a Conduché fibration.

Hence p_* exists and there is defined a polynomial functor

$$\begin{array}{ccc} \boldsymbol{Cat} & \longrightarrow & \boldsymbol{Cat} \\ C & \mapsto & p_*(C \times_{\mathbb{B}} \mathbb{E}) \end{array}$$

which we shall now describe in detail. The category $\mathbb{F}(C) := p_*(C \times_{\mathbb{B}} \mathbb{E})$ will be the category of families in C.

15.3.2 Objects of $\mathbb{F}(C)$. According to our description in 15.1.8, the objects in $\mathbb{F}(C)$ lying over an object $I \in \mathbb{B}$ are diagrams like

$$\begin{tikzcd}
& C \times_{\mathbb{B}} \mathbb{E} \arrow[d] \\
\mathbb{E}_I \arrow[ur, "s"] \arrow[r, phantom, "\subset"] & \mathbb{E}
\end{tikzcd}$$

so it is just any set map $\mathbb{E}_I \to C$. Note that the fibre $\mathbb{E}_I$ is canonically identified with I itself, so the objects of $\mathbb{F}$ over I are maps

$$\iota : I \to C.$$

We also think of such a map as a family $(C_i \mid i \in I)$.

15.3.3 Arrows of $\mathbb{F}(C)$. To describe the arrows lying over some bijection $\varphi : I \xrightarrow{\sim} J$, we first describe the full subcategory $\mathbb{E}_\varphi \subset \mathbb{E}$. The object set of $\mathbb{E}_\varphi$ is the disjoint union of $\mathbb{E}_I$ and $\mathbb{E}_J$. In other words,

$$\text{obj}(\mathbb{E}_\varphi) = I \sqcup J.$$

The (non-identity) arrows in $\mathbb{E}_\varphi$ must lie over φ: this means that they are arrows in $\mathbb{E}$ going from an object of $\mathbb{E}_I$ to an object in $\mathbb{E}_J$. Now the objects in $\mathbb{E}_I$ and $\mathbb{E}_J$ are just the elements in I and J, and the arrows from an object in $\mathbb{E}_I$ to an object in $\mathbb{E}_J$ lying over φ is just to give a pair $(i, j) \in I \times J$ such that $\varphi(i) = j$. In other words, the set of all the arrows over φ are just the correspondence between the elements in I and J expressed by the bijection φ. (This is to say that for a given $\varphi : I \to J$ and given an object in the fibre over J, there is a unique overlying arrow—this is just the formulation of $\mathbb{E} \to \mathbb{B}$ being a discrete fibration.)

Now an arrow in $\mathbb{F}(C)$ lying over φ is a diagram

$$\begin{tikzcd}
& C \times_{\mathbb{B}} \mathbb{E} \arrow[d] \\
\mathbb{E}_{\varphi} \arrow[ur, "\Phi"] \arrow[r, phantom, "\subset"] & \mathbb{E}
\end{tikzcd}$$

which amounts to giving $\iota : I \to C$ and $\jmath : J \to C$ and a family of arrows

$$\Phi_i : \iota(i) \to \jmath(\varphi(i))$$

15.3.4 The category of families in C. In summary, $\mathbb{F}(C)$ is the *category of families of objects in* C: The objects are functors $\iota : I \to C$ where I is a finite set, and where an arrow from (I, ι) to $(J, \jmath)$ is a pair (φ, Φ) consisting of a bijection $\varphi : I \overset{\sim}{\to} J$ and a family $\Phi = (\Phi_i \mid i \in I)$ of arrows $\Phi_i : \iota(i) \to \jmath(\varphi(i))$.

The category $\mathbb{F}(C)$ can also be described as the *free symmetric monoidal category on* $\mathscr{C}$. THIS OUGHT TO BE EXPLAINED HERE.

upoid-of-trees

15.3.5 Trees. Now the category of trees $\mathbb{T}$ is the least fixpoint for the functor

$$\begin{array}{rcl}
\boldsymbol{Poly}(1) & \longrightarrow & \boldsymbol{Poly}(1) \\
Q & \longmapsto & \mathsf{Id} + P \circ Q
\end{array}$$

One checks that $\mathbb{T}$ is a groupoid.

!!!! $\mathbb{F} : \boldsymbol{Cat} \to (\boldsymbol{Cat}, +, 0)$ the free-finite-sums functor, which is left adjoint to the forgetful functor.!!!!!

Another construction, more like the free algebra construction, without the need of a category of polynomial functors: let F be an endofunctor of $\boldsymbol{Cat}$. An *F-category* is a category $\mathscr{C}$ equipped with a functor $F(\mathscr{C}) \to$

$\mathscr{C}$. The forgetful functor from F-categories to categories has a left adjoint (provided F is sufficiently nice, e.g. it should certainly be enough that it is a polynomial functor whose product part is given by a discrete fibration). Let $T : \textbf{\textit{Cat}} \to \textbf{\textit{Cat}}$ denote the corresponding monad. Then $T(1)$ is the groupoid of (non-planar) trees.

MANY THINGS TO INVESTIGATE: HOW ABOUT WEAK ENDOFUNCTORS? WEAK F-CATEGORIES?

The groupoid of trees is not rigid. I.e. between two trees there can be many different isomorphisms.

Now if we pass to P-trees for some polynomial endofunctor (monad?) then we do get a rigid groupoid of P-trees. The decoration involves specified bijections between the set of children of each node and some fixed set!

The category of (non-planar) constellations is the least fixpoint for the functor

$$\begin{array}{rcl} \textbf{\textit{Cat}}/T1 & \longrightarrow & \textbf{\textit{Cat}}/T1 \\ X & \longmapsto & 1 + F(X) \end{array}$$

where F denotes the free symmetric operad on a categorical collection X.

MANY QUESTIONS TO ANSWER HERE: I don't think there is presently a truly rigorous treatment of fixed point for endofunctors of ***Cat***. If $F : \textbf{\textit{Cat}} \to \textbf{\textit{Cat}}$ is an endofunctor, how do we define F-***Cat***, the category of F-objects? A morphism of F-objects should involve an (invertible) 2-cell. Suppose that two endofunctors $F : \textbf{\textit{Cat}} \to \textbf{\textit{Cat}}$ and $G : \textbf{\textit{Cat}} \to \textbf{\textit{Cat}}$ are connected by a natural transformation $a : F \to G$ which is an equivalence (I mean that the functor $a_X : F(X) \to G(X)$ is an equivalence of categories for every X). In this case, can we show that the least fixed point of F is equivalent to the least fixed point of G?.

15.4 Final functors and discrete fibrations

Chapter 16

[Polynomial functors for presheaf categories]

16.1 Some prelims

Kan extensions

A left Kan extension of some functor $f : \mathbb{C} \to \mathbb{D}$ along another functor $i : \mathbb{C} \to \overline{\mathbb{C}}$ is a functor $\operatorname{lan}_i f : \overline{\mathbb{C}} \to \mathbb{D}$ with a natural transformation

$$\begin{tikzcd}
\mathbb{C} \arrow[rr, "i"] \arrow[dr, "f"'] & \arrow[d, phantom, "\Rightarrow"] & \overline{\mathbb{C}} \arrow[dl, dashed, "\operatorname{lan}_i f"] \\
& \mathbb{D} &
\end{tikzcd}$$

with the following universal property: for any other natural transformation

$$\begin{tikzcd}
\mathbb{C} \arrow[rr, "i"] \arrow[dr, "f"'] & \arrow[d, phantom, "\Rightarrow"] & \overline{\mathbb{C}} \arrow[dl, "g"] \\
& \mathbb{D} &
\end{tikzcd}$$

there is a unique natural transformation $\operatorname{lan}_i f \Rightarrow g$ making the obvious pasting of 2-cells commute.

There is also the notion of pointwise left Kan extension: for each object $X \in \overline{\mathbb{C}}$, consider the lax pullback square

$$\begin{tikzcd}
\mathbb{C}{\downarrow}X \arrow[r] \arrow[d] \arrow[dr, phantom, "\Rightarrow"] & 1 \arrow[d, "\ulcorner X \urcorner"] \\
\mathbb{C} \arrow[r, "i"'] & \overline{\mathbb{C}}
\end{tikzcd}$$

Now ask that the pasting

$$\begin{tikzcd}
\mathbb{C}{\downarrow}X \arrow[r] \arrow[d] \arrow[dr, phantom, "\Rightarrow"] & 1 \arrow[d, "\ulcorner X \urcorner"] \\
\mathbb{C} \arrow[r, "i"'] \arrow[dr, "f"'] \arrow[dr, phantom, "\Rightarrow", shift left=2] & \overline{\mathbb{C}} \arrow[d, "\mathrm{lan}_i f"] \\
 & \mathbb{D}
\end{tikzcd}$$

have the universal property.

If something is a pointwise left Kan extension then it is also a left Kan extension. This is true in ***Cat*** – note that all these notions make sense in any 2-category. If $\overline{\mathbb{C}}$ is cocomplete then the two notions agree.

Check out Mac Lane. . .

Categories of elements

16.1.1 Introduction. The elements of a set are just of one kind, and that's what we usually understand by elements. For a graph, there are two kinds of elements: vertices and edges, and these two kinds of elements are related by certain incidence relations. These considerations make sense for any presheaf category: for each object of a presheaf category there is a category of elements which is important in many constructions. (Recall that a graph (by which we mean an oriented, non-reflexive graph) is just a presheaf on the category $\mathbb{G}_1 = \{0 \rightrightarrows 1\}$.)

Let $\mathbb{C}$ be a small category, and let $\widehat{\mathbb{C}} := \boldsymbol{PrSh}(\mathbb{C}) := \boldsymbol{Fun}(\mathbb{C}^{\mathrm{op}}, \boldsymbol{Set})$ denote the category of presheaves on $\mathbb{C}$. We have the Yoneda embedding

$$\begin{aligned} y : \mathbb{C} &\longrightarrow \widehat{\mathbb{C}} \\ C &\longmapsto \mathrm{Hom}(\_, C). \end{aligned}$$

The Yoneda embedding is fully faithful, and we just write C for $y(C)$ unless there is a special reason to distinguish the two.

Recall the Yoneda Lemma: for $F \in \widehat{\mathbb{C}}$, there is a natural bijection

$$F(C) = \mathrm{Hom}(C, F)$$

Think of F as a geometric object assembled together from basic building blocks, namely the objects of $\mathbb{C}$. Each object C of $\mathbb{C}$ is thought of as a generic shape, or a generic figure. They are generic in the sense that they can realised in any F: a C-figure in F is by definition a morphism

$$C \to F$$

in $\widehat{\mathbb{C}}$. It is also called an element of F of shape C. In other words, an element in the set $F(C)$.

The elements of F assemble into a category denoted $\mathrm{el}(F)$. The objects are arrows $C \to F$ where C is an object of $\mathbb{C}$, and a morphism from $C \to F$ to $C' \to F$ is an arrow $C \to C'$ in $\mathbb{C}$ such that the triangle commutes:

$$\begin{tikzcd}
C \arrow[rr] \arrow[dr] & & C' \arrow[dl] \\
& F &
\end{tikzcd}$$

In other words, $\mathrm{el}(F)$ is just the comma category y/F, which in turn could be defined as the 2-pullback

$$\begin{tikzcd}
y/F \arrow[r] \arrow[d] \arrow[dr, phantom, "\Rightarrow"] & 1 \arrow[d, "\ulcorner F \urcorner"] \\
\mathbb{C} \arrow[r, "y"'] & \widehat{\mathbb{C}}
\end{tikzcd}$$

16.1.2 Example. Let F be a graph, i.e. a presheaf on $\mathbb{G}_1 = \{0 \rightrightarrows 1\}$. Hence the set of vertices of F is $F_0 = F(0)$ which we see as the set of presheaf maps $0 \to F$, and the set of edges of F is $F_1 = F(1)$ which we see as the set of presheaf maps $1 \to F$. So the set of objects of $\mathrm{el}(F)$ is $F_0 + F_1$. What is a morphism in $\mathrm{el}(F)$? Well, by definition, these are arrows in $\mathbb{G}_1$

compatible with the elements. So for example, to have an arrow

$$\begin{tikzcd}
0 \arrow[rr, "s"] \arrow[dr, "v"'] & & 1 \arrow[dl, "e"] \\
& F &
\end{tikzcd}$$

from vertex v to edge e is to say that v is the source vertex of e. In other words, for each edge $1 \to F$ there are two morphisms in $\mathrm{el}(F)$ ending there: one coming from the source of the edge, and one coming from the target of the edge. So we can think of the category of elements of a graph as its barycentric subdivision, with all the half-edges oriented towards the midpoint.

(If we do not want to take this highly Yoneda viewpoint, an element is a pair (C, e) where C is an object of $\mathbb{C}$, and e is an element of $F(C)$. In the present case, there is a morphism $(0, v) \to (1, e)$ for each $\alpha : 0 \to 1$ such that $\alpha^* e = v$.)

16.1.3 Example. If C is a representable object of $\widehat{\mathbb{C}}$, then we have

$$\mathrm{el}(C) = \mathbb{C}/C,$$

as it easily follows from the Yoneda lemma.

16.1.4 Remark. The natural projection functor $\mathrm{el}(F) \to \mathbb{C}$ is a discrete fibration. Conversely, given a discrete fibration $\mathbb{F} \to \mathbb{C}$ we get a presheaf by associating to an object $C \in \mathbb{C}$ its fibre. This gives an equivalence of categories between ***PrSh***$(\mathbb{C})$ and the category of discrete fibrations over $\mathbb{C}$.

The main property of the category of elements of F is that it is a canonical recipe for describing F as a colimit of generic figures. More precisely:

16.1.5 Theorem. *The composite functor*

$$\mathrm{el}(F) \to \mathbb{C} \to \widehat{\mathbb{C}}$$

has colimit F.

See Reyes-Reyes-Zolfaghari [93], Theorem. 4.2.2.

Proof. The functor takes an element $e : C \to F$ to the representable C. It is clear that each such image object has a map to F, namely e, and that all these maps are compatible. Hence F is a cocone. Let now Q be any other cocone. This means that for each $e : C \to F$ (object in the indexing category) we have a map $\Phi_e : C \to Q$ in $\widehat{\mathbb{C}}$ compatible with the arrows in $\mathrm{el}(F)$. This means that for each diagram

$$\begin{tikzcd} C \arrow[rr] \arrow[dr, "e"'] & & C' \arrow[dl, "e'"] \\ & F & \end{tikzcd} \quad \text{we have} \quad \begin{tikzcd} C \arrow[rr] \arrow[dr, "\Phi_e"'] & & C' \arrow[dl, "\Phi_{e'}"] \\ & Q & \end{tikzcd}$$

in $\widehat{\mathbb{C}}$. Now it is easy to see there is a unique map $F \to Q$: pointwise there is no choice but one: at the object $C \in \mathbb{C}$, our map $F(C) \to Q(C)$ must be given by sending $e : C \to F$ to $\Phi_e : C \to Q$. On arrows: given $\alpha : C \to C'$, we have $\alpha* : F(C') \to F(C)$ given by sending $e' : C' \to F$ to the composite $C \to C' \to F$. We send this to the composite $C \to C' \to Q$, which is well-defined because of the cocone compatibility condition on Q. It is clearly the only way to define the value on objects and arrows of $F \to Q$, and it is clearly natural (i.e. defines a morphism in the category of presheaves). Hence there is a unique morphism $F \to Q$ □

Try to use the notation

$$\begin{tikzcd} y{\downarrow}F \arrow[r] \arrow[d] \arrow[dr, phantom, "\Rightarrow"] & 1 \arrow[d, "\ulcorner F \urcorner"] \\ \mathbb{C} \arrow[r, "y"'] & \widehat{\mathbb{C}} \end{tikzcd}$$

For the canonical cocone (which we claim is a colimit cocone). The claim is that for any other cocone

$$\begin{tikzcd} y{\downarrow}F \arrow[r] \arrow[d] \arrow[dr, phantom, "\underset{\Phi}{\Rightarrow}"] & 1 \arrow[d, "\ulcorner Q \urcorner"] \\ \mathbb{C} \arrow[r, "y"'] & \widehat{\mathbb{C}} \end{tikzcd}$$

there is a unique 2-cell $\ulcorner F \urcorner \Rightarrow \ulcorner Q \urcorner$. That is, a unique map $F \to Q$ in $\widehat{\mathbb{C}}$. This is just to say, for each $e : C \to F$ an element of Q, but this element has to be $\Phi_e : C \to Q\ldots$

16.1.6 Proposition. *For any presheaf $F \in \widehat{\mathbb{C}}$ there is a natural equivalence*

$$\widehat{\mathbb{C}}/F \simeq \boldsymbol{PrSh}(\mathrm{el}(F)).$$

Remark: for a representable, this just says that $\widehat{\mathbb{C}}/C \simeq \widehat{\mathbb{C}/C}$.

Proof. Given a presheaf Q on $\mathrm{el}(F)$, consider its category of elements

$$\mathrm{el}(Q) \to \mathrm{el}(F) \to \mathbb{C}$$

These are discrete fibrations, and in particular the first functor is a cartesian fibred functor, and hence corresponds to a presheaf G over F.

Conversely, given a presheaf G over F, considering it as a morphism in $\widehat{\mathbb{C}}$ we apply the functor el to get $\mathrm{el}(G) \to \mathrm{el}(F) \to \mathbb{C}$. Since the composite and the second part are discrete fibrations, so is the first part, hence corresponds to a presheaf Q on $\mathrm{el}(F)$. □

All this is sort of abstract. Here is a direct construction using fibres and sums:

Proof. Given a presheaf $X : \mathbb{C}^{\mathrm{op}} \to \boldsymbol{Set}$ over F with structure map $p : X \to F$, define a presheaf

$$\begin{array}{rcl} Q : \mathrm{el}(F)^{\mathrm{op}} & \longrightarrow & \boldsymbol{Set} \\ \big[e : C \to F\big] & \longmapsto & \{x : C \to X \mid x.p = e\}. \end{array}$$

In other words, it is the set of liftings

$$\begin{tikzcd} & X \arrow[d, "p"] \\ C \arrow[ur, dashed] \arrow[r, "e"'] & F \end{tikzcd}$$

In other words still, we are sending $e : C \to F$ to the fibre of $p(C) : X(C) \to F(C)$ over e.

Conversely Given a presheaf $Q : \mathrm{el}(F)^{\mathrm{op}} \to \textbf{\textit{Set}}$, that means that for each $e : C \to F$ we have a set $Q(C, e)$. Now define a presheaf

$$\begin{array}{rcl} \mathbb{C}^{\mathrm{op}} & \longrightarrow & \textbf{\textit{Set}} \\ C & \longmapsto & \displaystyle\sum_{e:C \to F} Q(C, e). \end{array}$$

Clearly this comes equipped with a map of presheaves to F. [Check functoriality — that's routine.]

□

16.1.7 Proposition.

$$\operatorname{colim} F = \pi_0 \, \mathrm{el}(F).$$

(or is that only for covariant functors?)

Nerves

Let $F : \mathbb{C} \to \mathbb{D}$ be any functor. There is an associated 'nerve' functor

$$\begin{array}{rcl} N : \mathbb{D} & \longrightarrow & \widehat{\mathbb{C}} \\ D & \longmapsto & \mathrm{Hom}_{\mathbb{D}}(F-, D). \end{array}$$

These two functors are related by the Yoneda embedding and a natural transformation:

$$\begin{tikzcd}
\mathbb{C} \arrow[rr, "F"] \arrow[dr, "y"'] & {} \arrow[d, phantom, "\overset{\chi^F}{\Rightarrow}"] & \mathbb{D} \arrow[dl, "N"] \\
& \widehat{\mathbb{C}} &
\end{tikzcd}$$

whose components simply are the values of F on arrows: for fixed C we have a map of presheaves $\chi^F(C) : h_C \Rightarrow N(F(C))$, and at $X \in \mathbb{C}$ this amounts to $\mathrm{Hom}_{\mathbb{C}}(X, C) \to \mathrm{Hom}_{\mathbb{D}}(FX, FC)$.

Hence χ^F is invertible if and only if F is fully faithful.

See Weber [104] for more usage of such Yoneda structures.

16.1.8 Lemma. *N preserves all limits.*

Proof. Let $G : I \to \mathbb{D}$ be a functor admitting a limit in $\mathbb{D}$. Now $N(\lim G) = \lim(N \circ G)$ because we can compute pointwise:

$$N(\lim G)(C) = \mathrm{Hom}_{\mathbb{D}}(FC, \lim G) = \lim \mathrm{Hom}_{\mathbb{D}}(FC, G-) = \lim(N \circ G)(C).$$

□

16.1.9 Lemma. *Let D be an object of $\mathbb{D}$. Then the category of elements of ND is*

$$y{\downarrow}ND \simeq F{\downarrow}D.$$

Proof. This is a direct computation: the category of elements has as objects pairs (C, α) where $C \in \mathbb{C}$ and $\alpha \in ND(C) = \mathrm{Hom}_{\mathbb{D}}(FC, D)$. An arrow from (C', α') to (C, α) is given by an arrow $\phi : C' \to C$ in $\mathbb{C}$ such that $ND(\phi) : ND(C) \to ND(C')$ sends α to α'. But $ND(\phi)$ consists in precomposing with $F\phi$, so the condition is that this triangle commutes:

$$\begin{tikzcd}
FC' \arrow[rr, "F\phi"] \arrow[dr, "\alpha'"'] & & FC \arrow[dl, "\alpha"] \\
& D &
\end{tikzcd}$$

It is clear that the category we have just described is precisely $F{\downarrow}D$. □

The fundamental property of categories of elements is the following:

16.1.10 Proposition. *There is an equivalence of categories:*

$$\boldsymbol{PrSh}(y{\downarrow}F) \simeq \boldsymbol{PrSh}(C){\downarrow}F.$$

Given presheaf $Q : (y{\downarrow}F)^{\mathrm{op}} \to \boldsymbol{Set}$, define

$$\begin{array}{rcl} X : \mathbb{C}^{\mathrm{op}} & \longrightarrow & \boldsymbol{Set} \\ C & \longmapsto & \displaystyle\sum_{\alpha : C \to F} Q(C, \alpha). \end{array}$$

Clearly this presheaf has a natural map to F.

Conversely, given $X : \mathbb{C}^{\mathrm{op}} \to \boldsymbol{Set}$ with a natural transformation $X \Rightarrow F$, define

$$\begin{array}{rcl} Q : (y{\downarrow}F)^{\mathrm{op}} & \longrightarrow & \boldsymbol{Set} \\ [\alpha : C \to F] & \longmapsto & X(C)_{\alpha}, \end{array}$$

the fibre of $X(C) \to F(C) = \mathrm{Hom}_{\mathbb{C}}(C, F)$ over α.

If we take $F = ND$ and combine with the above lemma we get:

$$\boldsymbol{PrSh}(F\downarrow D) \simeq \boldsymbol{PrSh}(y\downarrow ND) \simeq \boldsymbol{PrSh}(C)\downarrow ND.$$

(In this result, if we take $\mathbb{D} = \mathbb{C}$ and take F to be Yoneda, then N is the identity functor, and we recover the basic result.)

Let us prove the equivalence

$$\boldsymbol{PrSh}(F\downarrow D) \simeq \boldsymbol{PrSh}(C)\downarrow ND$$

directly.

Given presheaf $Q : (F\downarrow D)^{\mathrm{op}} \to \boldsymbol{Set}$, define

$$\begin{array}{rcl} X : \mathbb{C}^{\mathrm{op}} & \longrightarrow & \boldsymbol{Set} \\ C & \longmapsto & \displaystyle\sum_{\alpha : FC \to D} Q(C, \alpha) \end{array}$$

We should also explain what X does on arrows: given $\phi : C' \to C$, let α' denote the composite

$$FC' \xrightarrow{F\phi} FC \xrightarrow{\alpha} D.$$

This is the unique way to make ϕ into an arrow in the category $F\downarrow D$. Now $Q(\phi)$ goes from $Q(C, \alpha)$ to $Q(C', \alpha')$, and we have defined $X(\phi)$. It is clear that X has a natural transformation to ND, hence we have constructed an object in $\boldsymbol{PrSh}(C)\downarrow ND$.

Conversely, given $X : \mathbb{C}^{\mathrm{op}} \to \boldsymbol{Set}$ with a natural transformation $X \Rightarrow ND$, define

$$\begin{array}{rcl} Q : (F\downarrow D)^{\mathrm{op}} & \longrightarrow & \boldsymbol{Set} \\ [\alpha : FC \to D] & \longmapsto & X(C)_\alpha, \end{array}$$

the fibre of $X(C) \to ND(C) = \mathrm{Hom}_{\mathbb{D}}(FC, D)$ over α.

Generic morphisms

16.1.11 Generic maps The notion of generic map was introduced by Weber [103] extending the notion of [55]. What we call generic is what in [103] is called strict generic. Our terminology agrees with Weber [104].

Let $T : \mathbb{D} \to \mathbb{C}$ be a functor. An arrow $g : X \to TD$ in $\mathbb{C}$ is called *generic* (with respect to T) if for every commutative square

$$\begin{tikzcd}
X \arrow[r] \arrow[d, "g"'] & TC \arrow[d, "T(c)"] \\
TD \arrow[r, "T(b)"'] & TY
\end{tikzcd}$$

there exists a unique arrow $u : D \to C$ such that

$$\begin{tikzcd}
 & C \arrow[d, "c"] \\
D \arrow[ur, "u"] \arrow[r, "b"'] & Y
\end{tikzcd} \quad \text{and} \quad \begin{tikzcd}
X \arrow[r] \arrow[d, "g"'] & TC \\
TD \arrow[ur, "T(u)"'] &
\end{tikzcd}$$

The functor T is said to *admit generic factorisations* if every $A \to TY$ admits a factorisation into a generic map followed by one of the form $T(\ )$. Such a factorisation is then necessarily unique up to unique isomorphism.

16.1.12 Parametric right adjoints. A functor $T : \mathbb{D} \to \mathbb{C}$ is called a *parametric right adjoint* if for every object Y of $\mathbb{D}$, the induced functor

$$\begin{array}{ccc} \mathbb{D}/Y & \longrightarrow & \mathbb{C}/TY \\ [X \to Y] & \longmapsto & [TX \to TY] \end{array}$$

is a right adjoint.

Observe that if T is a right adjoint itself, then it is also a parametric right adjoint.

16.1.13 Proposition. *A functor $T : \mathbb{D} \to \mathbb{C}$ is a parametric right adjoint if and only if it admits generic factorisations.*

Proof. Fix an object Y of $\mathbb{D}$, and suppose every map $X \to TY$ has a generic factorisation (and suppose one such factorisation has been chosen). We will construct a left adjoint to

$$\begin{array}{ccc} \mathbb{D}/Y & \longrightarrow & \mathbb{C}/TY \\ [X \to Y] & \longmapsto & [TX \to TY] \end{array}$$

On objects, the left adjoint is given by sending $B \to TY$ to the map $D \to Y$ appearing in the generic factorisation $B \to TD \to TY$. □

If $T = U \circ F$ for a free-forgetful adjunction $F \dashv U$, we suppress U from the notation and call a map $FA \to FB$ generic if it corresponds to a generic map $A \to FB$ under adjunction. This leads to the notion of *generic/free factorisation system*.

FOR THIS TO BE TRUE, MUST REQUIRE THAT THE FREE MAPS CONTAIN ALL ISOMORPHISMS. THERE WILL BE A FACTORISATION SYSTEM IN ANY CASE, BUT ONE MAY NEED TO SATURATE THE CLASS OF FREE MAPS BY THE ISOS.

Recall that a functor $G : \mathscr{D} \to \mathscr{C}$ is a right adjoint if and only if for all objects $X \in \mathscr{C}$, the comma category $X{\downarrow}G$ has an initial object. (In this category, the objects are pairs (D, α) with $D \in \mathscr{D}$ and $\alpha : X \to GD$. An arrow from (D, α) to (D', α') is an arrow $\phi : D \to D'$ in $\mathscr{D}$ such that

$$\begin{tikzcd}
 & X \arrow[dl, "\alpha"'] \arrow[dr, "\alpha'"] & \\
GD \arrow[rr, "G\phi"'] & & GD'
\end{tikzcd}$$

End of reminder.)

Now a functor $G : \mathscr{D} \to \mathscr{C}$ is a *local right adjoint* if for each object $D \in \mathscr{D}$ the sliced functor

$$\begin{aligned} G_D : \mathscr{D}/D &\longrightarrow \mathscr{C}/GD \\ [A \to D] &\longmapsto [GA \to GD] \end{aligned}$$

is a right adjoint. In other words, for each $x : X \to GD$, the comma category $x{\downarrow}G_D$ has an initial object. (To spell out what this comma category is: its objects are pairs (m, α) where $m : M \to D$ is in $\mathscr{D}/D$ and $\alpha : x \to Gm$, so altogether an object is a triangle

$$\begin{tikzcd}
X \arrow[dd, "x"'] \arrow[dr, "\alpha"] & \\
 & GM \arrow[dl, "Gm"] \\
GD &
\end{tikzcd}$$

In other words, factorisations of $x : X \to GD$ into $X \overset{\alpha}{\to} GM \overset{Gm}{\to} GD$. Arrows in this category are the obvious maps of factorisations,

$$\begin{tikzcd}
& X \arrow[dl, "\alpha"'] \arrow[dr, "\alpha'"] & \\
GM \arrow[rr, "G\phi"'] \arrow[dr, "Gm"'] & & GM' \arrow[dl, "Gm'"] \\
& GD &
\end{tikzcd}$$

To say that each such category has an initial object is practically the same as having generic factorisations. To establish that those initial factorisations have indeed generic first part amounts to establishing a stronger universal property: to say it is initial refers only to maps with a fixed target GD. To say it is generic involves arbitrary targets... It is a bit tricky to show that the first-components in the initial factorisations are in fact generic.

The category of factorisations of $x : X \to GD$ is the comma category

$$\begin{tikzcd}
x{\downarrow}G_D \arrow[r] \arrow[d] \arrow[dr, phantom, "\Rightarrow"] & \mathscr{D}/D \arrow[d, "G_D"] \\
1 \arrow[r, "\ulcorner x \urcorner"'] & \mathscr{C}/GD.
\end{tikzcd}$$

This refers to a fixed x. We can consider a bigger category where x varies:

$$\begin{tikzcd}
(\mathscr{C}/GD){\downarrow}G_D \arrow[r] \arrow[d] \arrow[dr, phantom, "\Rightarrow"] & \mathscr{D}/D \arrow[d, "G_D"] \\
\mathscr{C}/GD \arrow[r, "\mathrm{id}"'] & \mathscr{C}/GD.
\end{tikzcd}$$

In the category $(\mathscr{C}/GD){\downarrow}G_D$, the objects are triples (x, m, α) with $x : X \to$

GD, $m : M \to D$ and $\alpha : x \to Gm$. In other words, they are triangles

$$\begin{tikzcd}
X \arrow[dd, "x"'] \arrow[dr, "\alpha"] & \\
& GM \arrow[dl, "Gm"] \\
GD &
\end{tikzcd}$$

with variable X. The morphisms are now diagrams

$$\begin{tikzcd}
X \arrow[rr, "\phi"] \arrow[d, "\alpha"'] & & X' \arrow[d, "\alpha'"] \\
GM \arrow[rr, "G\psi"] \arrow[dr, "Gm"'] & & GM' \arrow[dl, "Gm'"] \\
& GD &
\end{tikzcd}$$

(This is just like a category of arrows, except that some of the lower parts of the diagrams come from another category.)

This category is fibred over $\mathscr{C}$ by returning the X. The cartesian arrows are those (ϕ, ψ) for which ψ is an isomorphism. The fibre over an object X is precisely the original category of factorisations with fixed start vertex X.

Monads with arities

16.1.14 Monads with arities. (Cf. Weber [104].) A *monad with arities* on a category $\mathscr{C}$ (not required to be a presheaf category or even to have a terminal object) consists of a monad F (not required to be cartesian) and a full subcategory $i_0 : \Theta_0 \subset \mathscr{C}$ required to be dense and small, such that the following condition is satisfied: the left Kan extension

$$\begin{tikzcd}
\Theta_0 \arrow[rr, "i_0"] \arrow[dr, "i_0"'] & \overset{\mathrm{id}}{\Rightarrow} & \mathscr{C} \arrow[dl, dashed, "\mathrm{id}"] \\
& \mathscr{C} &
\end{tikzcd}$$

is preserved by the composite

$$\mathscr{C} \xrightarrow{F} \mathscr{C} \xrightarrow{N_0} \boldsymbol{PrSh}(\Theta_0)$$

(cf. the proof of Proposition 16.1.16 below).

As above, denote by Θ the Kleisli category of F restricted to Θ_0, i.e. the full subcategory consisting of algebras that are free on an object of Θ_0, as in this diagram:

$$\begin{tikzcd}
\Theta \arrow[r, "i"] & F\text{-}\boldsymbol{Alg} \arrow[d, shift left=2, "\text{forgetful}"] \arrow[d, phantom, "\dashv"] \\
\Theta_0 \arrow[u, "j"] \arrow[r, "i_0"'] & \mathscr{C}. \arrow[u, shift left=2, "\text{free}"]
\end{tikzcd}$$

and let $N : F\text{-}\boldsymbol{Alg} \to \boldsymbol{PrSh}(\Theta)$ and $N_0 : \mathscr{C} \to \boldsymbol{PrSh}(\Theta_0)$ be the nerve functors induced by i and i_0 respectively.

general-nerve

16.1.15 General nerve theorem. (Cf. Weber [104, Thm. 4.10].) *If (F, Θ_0) is a monad with arities, then N is fully faithful, and $X : \Theta^{\text{op}} \to$ **Set** is in the essential image of N if and only if its restriction to Θ_0 is in the essential image of N_0.*

The following theorem should also be attributed to Weber [104]. It is analogous to his Prop.4.22.

arities

16.1.16 Proposition. *Let F be a monad on an arbitrary category $\mathscr{C}$, and let $i_0 : \Theta_0 \to \mathscr{C}$ be fully faithful and dense (with Θ_0 small). If the functor $i_0{\downarrow}Fi_0{\downarrow}F \longrightarrow i_0{\downarrow}F$ has non-empty connected fibres then Θ_0 provides F with arities.*

Let us spell out what those comma categories are: the first category $i_0{\downarrow}Fi_0{\downarrow}F$ is the category of factorisations: the objects are quintuples $(\mathsf{A}, \mathsf{M}, \mathsf{P}, g, m)$ where $\mathsf{A} \in \Theta_0$, $\mathsf{M} \in \Theta_0$, $\mathsf{P} \in \mathscr{C}$, and $g : \mathsf{A} \to \mathsf{M}$ in Θ_0 and $m : i_0 M \to \mathsf{P}$ in $\mathscr{C}$. The arrows are diagrams

$$\begin{tikzcd}
i_0\mathsf{A} \arrow[r, "i_0(g)"] \arrow[d, "i_0(u)"'] & Fi_0\mathsf{M} \arrow[r, "Fm"] \arrow[d, "Fi_0(v)"'] & F\mathsf{P} \arrow[d, "F(w)"] \\
i_0\mathsf{A}' \arrow[r, "i_0(g')"'] & Fi_0\mathsf{M}' \arrow[r, "Fm'"'] & F\mathsf{P}'
\end{tikzcd}$$

where u, v, and w are arrows in $\mathscr{C}$.

The second category $i_0{\downarrow}F$ consists of triples $(\mathsf{A}, \mathsf{P}, f)$ where $\mathsf{A} \in \Theta_0$, $\mathsf{P} \in \mathscr{C}$, and $f : i_0\mathsf{A} \to F\mathsf{P}$. The functor

$$i_0{\downarrow}Fi_0{\downarrow}F \longrightarrow i_0{\downarrow}F$$

just returns the composite arrow (i.e. undoes the factorisation). To say that it has non-empty fibres means that every map $f : i_0\mathsf{A} \to F\mathsf{P}$ admits a factorisation, and connectedness means that all the possible factorisations are connected by arrows.

Proof. We need to establish that the left Kan extension

$$\begin{tikzcd}
\Theta_0 \arrow[rr, "i_0"] \arrow[dr, "i_0"'] & \overset{\mathrm{id}}{\Rightarrow} & \mathscr{C} \arrow[dl, dashed, "\mathrm{id}"] \\
& \mathscr{C} &
\end{tikzcd}$$

is preserved by the composite

$$\mathscr{C} \xrightarrow{F} \mathscr{C} \xrightarrow{N_0} \textbf{\textit{PrSh}}(\Theta_0).$$

We will show it is a pointwise extension, i.e. that for each $\mathsf{P} \in \mathscr{C}$, the left Kan extension

$$\begin{tikzcd}
\Theta_0/\mathsf{P} \arrow[r] \arrow[d] \arrow[dr, phantom, "\overset{\lambda}{\Rightarrow}"] & 1 \arrow[d, "\ulcorner P \urcorner"] \\
\Theta_0 \arrow[r] & \mathscr{C}
\end{tikzcd}$$

is preserved by $N_0 \circ F$.

The claim is that (for fixed $X \in \textbf{\textit{PrSh}}(\Theta_0)$) each natural transformation

$$\begin{tikzcd}
\Theta_0/\mathsf{P} \arrow[r] \arrow[d] \arrow[dr, phantom, "\overset{\phi}{\Rightarrow}"] & 1 \arrow[drr, "\ulcorner X \urcorner"] & & \\
\Theta_0 \arrow[r] & \mathscr{C} \arrow[r, "F"'] & \mathscr{C} \arrow[r, "N_0"'] & \textbf{\textit{PrSh}}(\Theta_0)
\end{tikzcd}$$

factors uniquely as

$$\begin{tikzcd}
\Theta_0/\mathsf{P} \arrow[r] \arrow[d] \arrow[dr, phantom, "\overset{\lambda}{\Rightarrow}"] & 1 \arrow[d, "\ulcorner \mathsf{P} \urcorner"'] \arrow[drrr, "\ulcorner X \urcorner"] \arrow[dr, phantom, "\underset{\psi}{\Rightarrow}"] & & \\
\Theta_0 \arrow[r] & \mathscr{C} \arrow[r, "F"'] & \mathscr{C} \arrow[r, "N_0"'] & \boldsymbol{PrSh}(\Theta_0).
\end{tikzcd}$$

We shall first spell out in detail what we know about ϕ. The component of ϕ at a P-element $a : i_0\mathsf{A} \to \mathsf{P}$ is a map of presheaves $\phi_a : N_0 F i_0 \mathsf{A} \to X$, i.e. for each object $\mathsf{T} \in \Theta_0$ a natural map

$$\phi_{a,\mathsf{T}} : \mathscr{C}(i_0\mathsf{T}, F i_0 \mathsf{A}) \to X(\mathsf{T}).$$

The ϕ_a are presheaf maps, which means they are natural in T: given an arrow $u : \mathsf{T}' \to \mathsf{T}$ in Θ_0 we have a naturality square

$$\begin{tikzcd}
\mathscr{C}(i_0\mathsf{T}, F i_0 \mathsf{A}) \arrow[r, "\phi_{a,\mathsf{T}}"] \arrow[d, "u*"'] & X(\mathsf{T}) \arrow[d, "u*"] \\
\mathscr{C}(i_0\mathsf{T}', F i_0 \mathsf{A}) \arrow[r, "\phi_{a,\mathsf{T}'}"'] & X(\mathsf{T}')
\end{tikzcd}$$

We record the formula we need

$$\phi_{a,\mathsf{T}'}(h \circ u) = u^*\big(\phi_{a,\mathsf{T}}(h)\big), \tag{16.1}$$ nat-u

for every $h : i_0\mathsf{T} \to F i_0 A$. Finally we shall need the naturality of ϕ in a: given an arrow in $\Theta_0{\downarrow}\mathsf{P}$, i.e. a triangle

$$\begin{tikzcd}
i_0\mathsf{A} \arrow[rr, "i_0 q"] \arrow[dr, "a"'] & & i_0\mathsf{A}' \arrow[dl, "a'"] \\
& \mathsf{P} &
\end{tikzcd}$$

we have a naturality square

$$\begin{tikzcd}
N_0 F i_0 A \arrow[r, "\phi_a"] \arrow[d] & X \arrow[d, "\mathrm{id}"] \\
N_0 F i_0 A' \arrow[r, "\phi_{a'}"'] & X
\end{tikzcd}$$

hence for each T a commutative diagram

$$\begin{tikzcd}
\mathscr{C}(i_0\mathsf{T}, Fi_0A) \arrow[r, "\phi_{a,\mathsf{T}}"] \arrow[d, "(Fi_0q)_*"'] & X(\mathsf{T}) \arrow[d, "\mathrm{id}"] \\
\mathscr{C}(i_0\mathsf{T}, Fi_0A') \arrow[r, "\phi_{a',\mathsf{T}}"] & X(\mathsf{T})
\end{tikzcd}$$

We extract the equation we'll need:

$$\phi_{a,\mathsf{T}}(h) = \phi_{a',\mathsf{T}}(Fi_0q \circ h), \tag{16.2}$$

$\boxed{\texttt{nat-a}}$

for each $h : i_0\mathsf{T} \to Fi_0A$.

To specify $\psi : N_0F\mathsf{P} \to X$ we need for each object $\mathsf{T} \in \Theta_0$ a natural map

$$\psi_{\mathsf{T}} : \mathscr{C}(i_0\mathsf{T}, F\mathsf{P}) \to X(\mathsf{T}).$$

Finally, the component of $N_0 \circ F \circ \lambda$ at a P-element $a : i_0\mathsf{A} \to \mathsf{P}$ and an object $\mathsf{T} \in \Theta_0$ is

$$\begin{array}{rcl}
\mathscr{C}(i_0\mathsf{T}, Fi_0\mathsf{A}) & \longrightarrow & \mathscr{C}(i_0\mathsf{T}, F\mathsf{P}) \\
z & \longmapsto & Fa \circ z.
\end{array}$$

Now the key point is that every $f \in \mathscr{C}(\mathsf{T}, F\mathsf{P})$ is in the image of this map for a suitable a: since the category of factorisations of f is non-empty we can write a factorisation

$$\begin{tikzcd}
\mathsf{T} \arrow[rr, "f"] \arrow[dr, "g"'] & & F\mathsf{P} \\
& Fi_0\mathsf{M} \arrow[ur, "Fm"'] &
\end{tikzcd}$$

Now it is clear that we have

$$f = (N_0 \circ F \circ \lambda)_{m,\mathsf{T}}(g).$$

So if ψ is going to give ϕ after pasting with λ we are forced to define

$$\psi_{\mathsf{T}}(f) := \phi_{m,\mathsf{T}}(g) \in X(\mathsf{T}).$$

We claim that the value of ψ does not depend on which factorisation we choose for f. So suppose we have two factorisations of f. By connectedness of the category of factorisations we can assume the two factorisations are linked by an arrow:

$$\begin{tikzcd}
\mathsf{T} \arrow[r, "g'"] \arrow[d, "g"'] & F\mathsf{M}' \arrow[d, "Fm'"] \\
F\mathsf{M} \arrow[r, "Fm"'] \arrow[ur, dashed, "Fq"'] & F\mathsf{P}
\end{tikzcd}$$

Now we can compute

$$\phi_{m,\mathsf{T}}(g) = \phi_{m',\mathsf{T}}(Fq \circ g) = \phi_{m',\mathsf{T}}(g').$$

where the first equality was naturality (16.2) of ϕ in m. So in conclusion, our definition of ψ is independent of choice of factorisation, if just the space of factorisations is connected.

Now we should verify that ψ is in fact natural in T. So let there be given an arrow $u : \mathsf{T}' \to \mathsf{T}$ in Θ_0. We need to establish the commutativity of the square

$$\begin{tikzcd}
\mathscr{C}(i_0\mathsf{T}, F\mathsf{P}) \arrow[r, "\psi_{\mathsf{T}}"] \arrow[d, "u*"'] & X(\mathsf{T}') \arrow[d, "u*"] \\
\mathscr{C}(i_0\mathsf{T}', F\mathsf{P}) \arrow[r, "\phi_{a,\mathsf{T}'}"'] & X(\mathsf{T}')
\end{tikzcd}$$

So let us check this at some $f : i_0\mathsf{T} \to F\mathsf{P}$. Let $f' = f \circ u$. Factor f as g followed by Fm and factor f' as g' followed by Fm'. By connectedness of the category of factorisations we can assume there is a map $q : \mathsf{M} \to \mathsf{M}'$ connecting the two factorisations:

$$\begin{tikzcd}
\mathsf{T}' \arrow[r, "u"] \arrow[d, "g'"'] & \mathsf{T} \arrow[r, "g"] & F\mathsf{M} \arrow[d, "Fm"] \\
F\mathsf{M}' \arrow[rr, "Fm'"'] \arrow[urr, dashed, "Fq"'] & & F\mathsf{P}
\end{tikzcd}$$

Now we can compute

$$\phi_{m',\mathsf{T}'}(g') = \phi_{m\circ q,\mathsf{T}'}(g') \;=\; \phi_{m,\mathsf{T}'}(Fq\circ g') = \phi_{m,\mathsf{T}'}(g\circ u) \;=\; u^*(\phi_{m,\mathsf{T}}(g)),$$

where the second equality expresses naturality (16.2) of ϕ in m, and the last equality is naturality (16.1) of ϕ_m in T. □

16.2 Distributors and mixed fibrations

16.2.1 Spans. Let A and B be categories. An span from A to B is a diagram of categories

$$\begin{tikzcd} A & V \arrow[l, "p"'] \arrow[r, "q"] & B, \end{tikzcd}$$

and a morphism of spans is a diagram

$$\begin{tikzcd} & V \arrow[ld] \arrow[rd] \arrow[dd, "u"] & \\ A & & B \\ & W \arrow[lu] \arrow[ru] & \end{tikzcd}$$

Denote by $\boldsymbol{Span}(A,B)$ the category of spans from A to B.

16.2.2 Mixed fibrations. A *mixed fibration* from category $\mathbb{A}$ to category $\mathbb{B}$ is a span

$$\begin{tikzcd} & \mathbb{M} \arrow[ld, "q"'] \arrow[rd, "p"] & \\ \mathbb{A} & & \mathbb{B} \end{tikzcd}$$

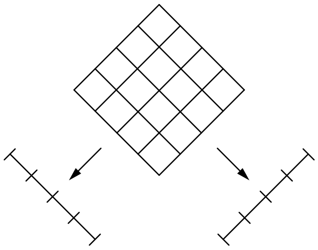
such that q is an opfibration, p is a fibration, and for each pair of objects $(A,B)\in\mathbb{A}\times\mathbb{B}$, the fibre

$$(q\times p)^{-1}(A,B)\subset\mathbb{M}$$

is discrete.

Are there more axioms to list, or are they consequences?

It follows that the opcartesian arrows for q are precisely the vertical arrows for p, and the cartesian arrows for p are precisely the vertical arrows for q. We should therefore think of a mixed fibration like this:



mixedfib-from-distr

16.2.3 Proposition. *Given a cospan (a, b) as in the diagram below, construct the comma square*

$$\begin{tikzcd}
& b{\downarrow}a \arrow[dl, "q"'] \arrow[dr, "p"] & \\
\mathbb{A} \arrow[dr, "a"'] & \Leftarrow & \mathbb{B} \arrow[dl, "b"] \\
& \S &
\end{tikzcd}$$

then the span (q, p) is a mixed fibration.

Proof. This should be a direct check. □

16.2.4 Left adjoint between presheaf categories, and distributors. (See Bénabou [16] for more details.) A *distributor* from $\mathbb{A}$ to $\mathbb{B}$ (also called an $(\mathbb{A}, \mathbb{B})$-*bimodule*) is a functor

$$M : \mathbb{B}^{\mathrm{op}} \times \mathbb{A} \to \boldsymbol{Set}.$$

Equivalently, it is a functor

$$\mathbb{A} \to \widehat{\mathbb{B}}.$$

By left Kan extension, this is equivalent to a cocontinuous functor

$$\widehat{\mathbb{A}} \to \widehat{B}.$$

We write it

$$\begin{tikzcd} \mathbb{A} \arrow[r, "|" marking] & \mathbb{B} \end{tikzcd}$$

In each case the morphisms are the natural transformations. Denote by $(\mathbb{A}, \mathbb{B})$-***Mod*** the category of $(\mathbb{A}, \mathbb{B})$-bimodules.

The cocontinuous-functor viewpoint tells us how distributors compose. That's precisely the tensor product of bimodules.

16.2.5 From spans to bimodules. A span (as above) defines a left adjoint

$$\widehat{\mathbb{B}} \xrightarrow{q^*} \widehat{\mathbb{V}} \xrightarrow{p_!} \widehat{\mathbb{A}},$$

and a morphism of spans as above defines a natural transformation which is essentially the counit $u_! \circ u^* \Rightarrow \mathrm{id}$. This defines a functor

$$L : \boldsymbol{Span}(\mathbb{A}, \mathbb{B}) \longrightarrow (\mathbb{A}, \mathbb{B})\text{-}\boldsymbol{Mod}.$$

16.2.6 Span representation for distributors. Given a distributor in the form of $\ell : \mathbb{A} \to \widehat{\mathbb{B}}$, we can form a span by the following comma square construction:

$$\begin{tikzcd}
& y{\downarrow}\ell \arrow[dl, "q"'] \arrow[dr, "p"] & \\
\mathbb{A} \arrow[dr, "\ell"'] & \Leftarrow & \mathbb{B} \arrow[dl, "y"] \\
& \widehat{\mathbb{B}} &
\end{tikzcd}$$

Then the left adjoint $L : \widehat{\mathbb{A}} \to \widehat{\mathbb{B}}$ is given as

$$L = q^* p_!$$

If $\theta : \ell \Rightarrow \ell'$ is a natural transformation between bimodules, the universal property of the comma object provides a span map $y{\downarrow}\ell \to y{\downarrow}\ell'$, so we have a functor

$$R : (\mathbb{A}, \mathbb{B})\text{-}\boldsymbol{Mod} \to \boldsymbol{Span}(\mathbb{A}, \mathbb{B}).$$

So any distributor can be represented by a mixed fibration.

16.2.7 Proposition. *The functors L and R form an adjoint pair*

$$L \dashv R,$$

*and the counit $L \circ R \Rightarrow \mathsf{Id}$ is an isomorphism. In other words, R is fully faithful and makes (A, B)-**Mod** a full reflexive subcategory of **Span**(A, B).*

Proof. Applying R to the bimodule $m : B \to \widehat{A}$ gives the span

$$A \xleftarrow{p} y{\downarrow}m \xrightarrow{q} B.$$

Applying now L gives $p_! \circ q^* \circ y_B$, and it is easy to check that this functor is naturally isomorphic to m. This accounts for the invertible counit of the adjunction.

To describe the unit we shall invoke the universal property of the comma object: starting from an arbitrary span

$$\begin{tikzcd} A & V \arrow[l, "p"'] \arrow[r, "q"] & B, \end{tikzcd}$$

we consider $L(V)$ as a functor $B \to \widehat{A}$, denoting it $m := p_! \circ q^* \circ y_B$. We shall exhibit a natural transformation

$$\begin{tikzcd} V \arrow[r, "q"] \arrow[d, "p"'] \arrow[dr, phantom, "\Nearrow"] & B \arrow[d, "m"] \\ A \arrow[r, "y_A"'] & \widehat{A}, \end{tikzcd}$$

then the universal property will give us a span map $V \to y{\downarrow}m$, which will be the V-component of the unit. For each $v \in V$ we need a natural transformation

$$y_A(pv) \Rightarrow mqv.$$

By Yoneda, this is equivalent to giving an element of the set

$$(mqv)(pv) = (p_! q^* y_B(qv))(pv) = (p^* p_! q^* y_B(qv))(v).$$

Since we have the unit $\mathsf{Id} \Rightarrow p^* p_!$, it is enough to give an element in

$$(q^* y_B(qv))(v) = B(qv, qv),$$

and in here we have of course the element $q(\mathrm{id}_v)$. This describes the unit of the claimed adjunction.

We omit the remaining verifications. □

16.2.8 **N**ote that a distributor can be represented by many different distributors. For example the identity functor can be represented by the identity span, but it can also be represented by the category of arrows (the comma cat $\mathrm{id} \downarrow \mathrm{id}$). Clearly these two spans are not isomorphic. Notice however that although the identity span looks like the canonical representation, in fact the category-of-arrows representation has an advantage: it is a mixed fibration. Clearly the identity span is not a mixed fibration!

Claim: although many different spans can represent the same distributor, there is only one mixed fibration that can represent it.

16.2.9 Remark. It follows that the subcategory of all spans inducing the same bimodule M is connected. In fact, since the unit for the adjunction is given by a universal property, this subcategory has a terminal object $R(M)$.

Here is an easy example to show that the unit is not an isomorphism, or even an equivalence of categories: consider the trivial span $A \leftarrow A \rightarrow A$. The corresponding bimodule is the 'trivial' bimodule, namely the one corresponding to the left adjoint id : $\widehat{A} \rightarrow \widehat{A}$, and the Yoneda embedding $y : A \rightarrow \widehat{A}$. Taking R on this bimodule we get

$$y{\downarrow}y,$$

which by the Yoneda lemma is easily seen to coincide with the category of arrows of A, clearly not equivalent to A itself (in general).

16.2.10 Remark. The reflection of a span via $R \circ L$ gives a notion of normal form for a span. This normal form is characterised by being a mixed fibration: this means that the left leg is a Grothendieck fibration, the right leg is a Grothendieck opfibration, and the functor $V \rightarrow A \times B$ has discrete fibres.

16.2.11 **I**nstead of invoking the Yoneda embedding, the exact same mixed fibration can be constructed using another important construction: the *join* category of a distributor. Perhaps it is rather called the *collage*. Let $\mathbb{A} \xrightarrow{\;\;|\;\;} \mathbb{B}$ be a distributor, given by $M : \mathbb{B}^{\mathrm{op}} \times \mathbb{A} \rightarrow \textbf{\textit{Set}}$. Construct a new category $J := \mathbb{B} \underset{M}{*} \mathbb{A}$:

this category J has as object set the disjoint union of the object sets of $\mathbb{B}$ and $\mathbb{A}$. The hom sets are

$$J(x,y) := \begin{cases} A(x,y) & \text{for } x,y \in A \\ B(x,y) & \text{for } x,y \in B \\ M(x,y) & \text{for } x \in B, y \in A \\ \emptyset & \text{for } x \in A, y \in B \end{cases}$$

Composition inside each of the two parts is given by the original compositions, and composition with the elements in M make sense by functoriality of M. We should think of M as being a left-$\mathbb{B}$-module and a right-$\mathbb{A}$-module.

Now if we consider the two inclusion functors and take the comma square we arrive at the same span as in the Yoneda embedding construction above.

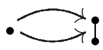
$$\begin{tikzcd}
 & j{\downarrow}i \arrow[dl, "q"'] \arrow[dr, "p"] & \\
\mathbb{B} \arrow[dr, "j"'] & \Rightarrow & \mathbb{A} \arrow[dl, "i"] \\
 & \mathbb{B} \underset{M}{*} \mathbb{A} &
\end{tikzcd}$$

16.3 The free-category monad

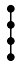
A key example of a polynomial functor so far has been the free-monoid monad on the category of sets. Moving beyond ***Set***, it is obvious to look at presheaf categories, and as a basic example we shall study the free-category monad on the category of (non-reflexive) directed graphs. We shall soon see that this monad is not polynomial in the sense of slices and locally cartesian closed categories, but that it is polynomial in the presheaf sense, where lowershriek and lowerstar denote the left and right Kan extensions to the pullback functor.

Observe that we already (8.7) described the free-category monad in the fixed object case. In a sense that accounts for everything, because of course the free category on a graph with vertex set S will have object set S again. However, that construction which we analysed in terms of slices, is very different from the construction we shall now perform.

Let $\mathbb{G}$ denote the category $0 \rightrightarrows 1$. Here is a picture of $\mathbb{G}$:



A *(non-reflexive) directed graph* is a presheaf on $\mathbb{G}$. In this section we just say *graph*, but beware that later on we shall consider more general notions of graphs. Every small category has an underlying graph whose vertices are the objects and whose edges are the arrows. The forgetful functor $\boldsymbol{Cat} \to \boldsymbol{PrSh}(\mathbb{G})$ has a left adjoint, the free-category functor: the free category on a graph G has the vertices of G as objects, and the paths in G as arrows. A *path* is just a map of graphs from a linear graph into G, and source and target of a path are given as the images of the endpoints of the linear graphs. Linear graphs are pictured as



Particularly important for us is the free category on the terminal graph. The terminal graph 1 is a single loop: it has one vertex an one edge (necessarily a loop on the unique vertex). The paths in 1 have no choice but running around in the loop, so there is one path for each $n \in \mathbb{N}$, counting how many time it winds around. So $T1$ is the category with only one object and $\mathbb{N}$ as set of arrows. Clearly composition of arrows is just addition in $\mathbb{N}$, so altogether, $F1$ is the monoid $\mathbb{N}$, considered as a category.

In complete generality we can factor T as

$$\begin{tikzcd}
\widehat{\mathbb{G}} \arrow[rr, "T"] \arrow[dr] & & \widehat{\mathbb{G}} \\
& \widehat{\mathbb{G}}/T1 \arrow[ur] &
\end{tikzcd}$$

Recall the fundamental isomorphism of categories $\widehat{\mathbb{G}}/T1 \simeq \boldsymbol{PrSh}(\mathrm{el}(T1))$ and that the forgetful functor $\widehat{\mathbb{G}}/T1 \to \widehat{\mathbb{G}}$ then becomes identified with $t_!$, where $t : \mathrm{el}(T1) \to \mathbb{G}$ is the discrete fibration corresponding to the fact that $T1$ is a presheaf on $\mathbb{G}$. Hence we have already found the last leg of the bridge diagram that is going to represent T.

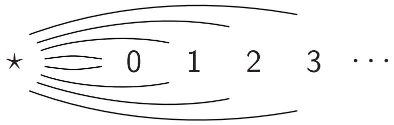
The category of elements $\mathbb{B} := \mathrm{el}(T1)$ has the following explicit description: there is one object $\star$ corresponding to the unique 0-element of $T1$ (the vertex), and then there is one object n for each $n \in \mathbb{N}$, corresponding to the 1-elements of $T1$. Finally, for each n there are two arrows $\star \to \mathsf{n}$ (source and target). The projection $t : \mathrm{el}(T1) \to \mathbb{G}$ sends $\star$ to 0 and all the other objects to 1. We should be explicit in deriving the arrows in $\mathrm{el}(T1)$: the elements are maps of presheaves

$$\mathrm{h}_k \to T1$$

(where h_k is the representable presheaf represented either by 0 or by 1. The arrows are commutative triangles of presheaf maps

$$\begin{tikzcd}
\mathrm{h}_0 \arrow[rr] \arrow[dr, "\star"'] & & \mathrm{h}_1 \arrow[dl, "\mathsf{n}"] \\
& T1 &
\end{tikzcd}$$

where we have already substituted 0 and 1 for k, because all the maps in $\mathbb{G}$ are like that: there are two maps $0 \rightrightarrows 1$. The claim is that in each case both these maps are valid (i.e. make the triangle commute). This we can check pointwise, and it is easy: a level 0 we have that $T1(0)$ is singleton, so every triangle ending here will commute. At level 1 we have $\mathrm{h}_0(1) = \emptyset$, so every triangle starting here will commute. So in $\mathbb{B}$ there will be the 'same maps' as in $\mathbb{G}$, but spread out and copied as necessary, just like in this figure:

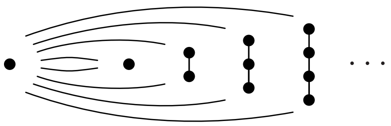


Now the way we described T in terms of paths means that the 1-elements in $T1$ are actually images of linear graphs — it is only winded up at a single vertex although it was constructed as a linear graph. We can construct a functor

$$E_T : \mathbb{B} = \mathrm{el}(T1) \to \widehat{\mathbb{G}}$$

which 'unwinds' each element — i.e. remembers it path shape. Hence E_T maps $\star$ to the terminal graph, and it maps each n to the linear graph with n edges. It is clear where the morphisms go: each pair $\star \rightrightarrows \mathsf{n}$ is sent to the inclusion of the terminal graph into the first and the last vertex of the length-n linear graph — that was the intention with our definition of free category that the new set of edges (arrows) between two given objects should be the start and finish of the path.

In fact we will take the liberty of already building this description of E_T into our picture of $\mathrm{el}(T1)$:



(This drawing is a mixture of the category $\mathrm{el}(T1)$ and its image in $\widehat{\mathbb{G}}$: it is in fact a picture of $\mathrm{el}(T1)$ but the objects are drawn in terms of their images in $\widehat{\mathbb{G}}$.)

The projection to $\mathbb{G}$ is obvious. Note that in $\mathbb{B}$ there are two maps from $\star$ to 0, this may seem strange. The formal answer is that if you actually compute the category of elements, this is what you find. Some more pragmatic answers are: there have to be two arrows in order for the projection to $\mathbb{G}$ to be a fibration! And finally, while $\star$ represents an object, 0 will represent an arrow, and an arrow has both a source and a target (even an identity arrow has both source and target).

The left Kan extension of E_T we denote

$$L : \widehat{\mathbb{B}} \to \widehat{\mathbb{G}}.$$

By construction (theory of distributors) this is cocontinuous, hence is a left adjoint. We can give a span representation of this functor

$$\begin{tikzcd}
 & \mathbb{E} \arrow[dl, "s"'] \arrow[dr, "p"] & \\
\mathbb{G} & & \mathbb{B}
\end{tikzcd}$$

Now

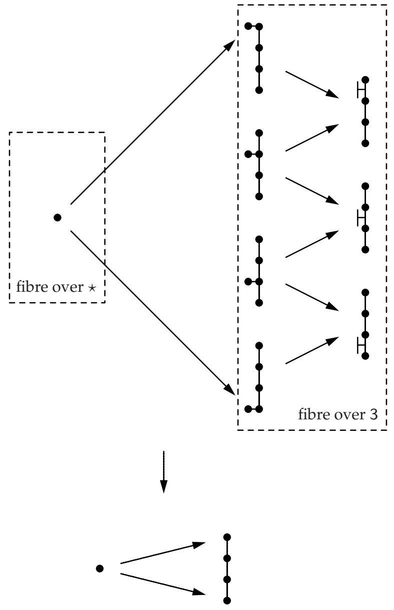
$$L = p^* \circ s_! \, .$$

The corresponding right adjoint $R = s^* p_*$ is what we need: it should now not be too difficult to show that $T = s^* p_* t_!$, so that the free-category monad T is presheaf polynomial, represented by

$$\begin{tikzcd}
 & \mathbb{E} \arrow[r, "p"] \arrow[dl, "s"'] & \mathbb{B} \arrow[dr, "t"] & \\
\mathbb{G} & & T & \mathbb{G}
\end{tikzcd}$$

So the whole functor is captured by giving it value on 1, and then the *spine functor* $E_T : \text{el}(T1) \to \widehat{\mathbb{G}}$. This is what characterises local right adjoints between presheaf categories. The check that in this case the construction does indeed give the free-category monad is the proof that the free-category monad is a local right adjoint — and in particular presheaf polynomial.

The category $\mathbb{E}$ has a explicit description, which is the key to see that the polynomial functor defined is really the free-category monad. To construct it, use distributor theory: E_T defines a distributor from $\mathbb{B}$ to $\mathbb{G}$, and we construct the canonical span representation (a mixed fibration). By the general construction 16.2.3, the middle object $\mathbb{E}$ is a sort of a variable category of elements of $\mathbb{B}$. This means that for each object $B \in \mathbb{B}$, the fibre $\mathbb{E}_B$ is the category of elements of $E_T(B)$. So the fibre over $\star$ is the category of elements of the graph with one vertex and no edges. This category is the terminal category. The fibre over n is the category whose objects the linear graphs n with one marked vertex or one marked edge. There are maps from the vertex-marked linear graphs to the edge-marked graphs, so that the marked vertex go into the endpoints of the marked edge. Here is a picture of the fragment of $\mathbb{E}$ lying over $\star \rightrightarrows 3$:



The fibre over $\star$ is $\star$, and the opcartesian lifts are the inclusions of $\star$ into the endpoint-vertex marked graphs.

On the other hand, the projection to $\mathbb{G} = 0 \rightrightarrows 1$: we know that the vertical maps for the projection to $\mathbb{B}$ should be the cartesian map for the projection to $\mathbb{G}$.

The projection to $\mathbb{B}$ sends $\star$ and all the vertex-marked graphs to 0, and it sends all the edge-marked graphs to 1. This is a fibration: the cartesian arrows are the inclusions of the vertex-marked graphs into the adjacent edge-marked ones.

Note that each object in $\mathbb{E}$ is really a morphism from some representable graphs to a linear graph (and then there is this extra copy of $\star$). If we look at the fibre over n and then take the image of that fibre down in $\mathbb{G}$ we find precisely the canonical diagram of n, i.e. its canonical expression as a colimit of representables.

This is also how we go back and forth between the span representation and E_T: We claimed that the left Kan extension of E_T was $p_! \circ q^*$. So its value on a representable n is: take colimit over the fibre over n.

Perhaps it is helpful to perform a example computation: how do we know that the maps in $\mathbb{E}$ out of $\star$ are precisely those two we claimed?

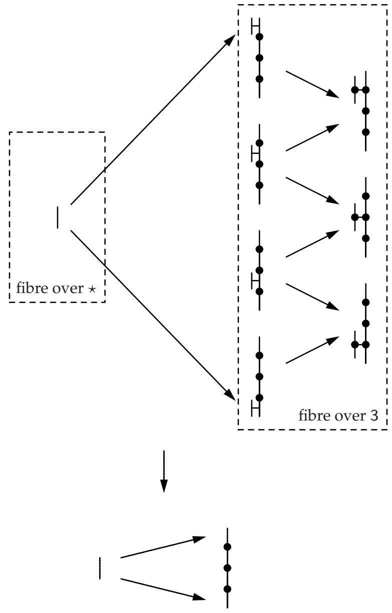
Remember that $\mathbb{E}$ is constructed so that its objects are triples (g, b, α) where g is an object in $\mathbb{G}$, b is an object in $\mathbb{B}$, and $\alpha : g \to b$ is an arrow in $\widehat{G}$ (really from $y(g)$ to $E_T(b)$).

Well the object $\star$ is really $\star_{\mathbb{G}} \to \star_{\mathbb{B}}$ (it's a $\star$-shaped element of the second star), while the vertex-marked graphs are really $\alpha : \star_{\mathbb{G}} \to \mathsf{n}$ (a $\star$-shaped element of n). By definition of morphism in the comma category, we must give a map $\star \to \star$ in $\mathbb{G}$ (this has to be the identity of course) together with a map $\star \to \mathsf{n}$ in $\mathbb{B}$ such that the obvious diagram in $\widehat{G}$ commutes. But this means that the map α must necessarily be one of the two maps that exist in $\mathbb{B}$ from $\star$ to n, and these are the endpoint inclusions.

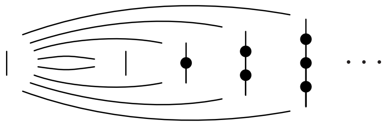
Another way to construe $\mathbb{E}$ is as the twisted category of elements of E_T.

Now that we have understood the construction in the viewpoint of directed graphs and categories, notice that although we talk about graphs and categories, the pictures do not involve any graphs more complicated than the linear ones. These are the building blocks, and the whole functor is expressed at this level, without passing to presheaves (the graphs). Now, to anticipate the next example, the free-multicategory monad, let us change the pictorial interpretation of the categories $\mathbb{G}$, $\mathbb{B}$, and $\mathbb{E}$: instead of a dot we draw a line segment (a string), and instead of the edge between two dots we draw a one-dot linear tree — a dot sitting between two strings.

Here is the new picture of the fragment of $\mathbb{E}$ sitting over $\star \rightrightarrows 3$:



The picture of $\mathbb{B}$ is:



The free-multicategory monad

Now we are ready to generalise this to the free-multicategory monad. We are now considering presheaves on the category of elementary planar trees. So here we have an object $\star$ which is the trivial tree, and then we have a tree n for each $n \in \mathbb{N}$. For each n we have one map from $\star$ corresponding to the output edge of n, and further n maps $\star \to \mathsf{n}$ corresponding to the input edges.

We are first interested in the free multicategory on the terminal presheaf. This is again a presheaf, and its value on $\star$ is singleton, and its value on n

is the set of all planar trees with n leaves. So we can calculate its category of elements

$$\mathbb{B} := \mathrm{el}(T1) :$$

It's object set is the disjoint union of all the sets we just described: so its object set is the set of all planar trees, together with one special copy of the trivial tree which we denote $\star$. Arguing exactly like in the free-category case, it is easy to see that the arrows in $\mathbb{B}$ are these: hey all have $\star$ as domain, and for each tree there t there is one arrow $\star \to t$ (corresponding to the output edge), and furthermore n maps $\star \to t$ if t has n leaves. Note that there are two maps from $\star$ to the trivial tree!

The description of the elements of $T1$ as trees already anticipates the spine functor $E_T : \mathbb{B} \to \widehat{\mathbb{I}}$. Using this, we can no easily construct $\mathbb{E}$: the fibre over a tree $t \in \mathbb{B}$ is the set of its elements: i.e. one object for each inclusion of an edge into t, and one object for each inclusion of a one-dot tree into t. The fibre over the special object $\star \in \mathbb{B}$ is singleton, and we denote it $\star$ again. There are only a few arrows in $\mathbb{E}$ from $\star$: namely one arrow to an edge-marked tree if the marked edge is a leaf and one arrow to an edge marked tree if the marked edge is the root edge. Note that this description was carefully formulated so as to give *two* arrows in the special case of the edge-marked trivial tree!

Altogether: $\mathbb{I}$ is the category of elementary planar trees.

$\mathbb{B}$ is the category whose objects are the planar trees, together with one special object $\star$, and whose morphisms are one from $\star$ to a tree for each of its leaves, and one from $\star$ to a tree for its root edge.

$\mathbb{E}$ is the category whose objects are maps $e \to t$ where e is an elementary tree and t is a tree. The arrows between these objects are the commutative triangles: i.e. maps between elementary trees commuting with the maps to a fixed tree. In addition to all that, there is one special object $\star$, and an arrow from this to each leaf-marked tree, and also to each root-marked tree. (This makes two arrows from $\star$ to the trivial tree (with the unique edge marked.) So the free-multicategory monad is presheaf polynomial, given by

$$\begin{tikzcd}
 & \mathbb{E} \arrow[r, "p"] \arrow[dl, "s"'] & \mathbb{B} \arrow[dr, "t"] & \\
\mathbb{I} & & T & \mathbb{I}
\end{tikzcd}$$

Note that in this case t is a discrete fibration: in the fibre over an elementary tree n, we have all the trees with n leaves, but no morphisms between them. So the free-multicategory monad is a local right adjoint.

The free-coloured-operad monad

Also called free-symmetric-multicategory monad.

The pictures are mostly the same as in the previous case, but care is required with the morphisms, since there are symmetries to keep track of.

Already $\mathbb{I}$ is more complex since it has for each n, a whole groupoid $\mathfrak{S}_n$ of automorphisms. But the main difference comes in the description of $\mathbb{B}$: it is no longer a discrete fibration over $\mathbb{I}$. This is because the fibre over an elementary tree n is still the category of trees with n leaves, and morphisms that fix those leaves, but since we have nullary operations at our disposal, there are still automorphisms inside the fibre.

So, $\mathbb{B}$ is the category whose objects are trees, and with one special extra object $\star$. The maps are: all the isomorphisms of trees. And also maps from $\star$ to trees precisely as before.

Now the category $\mathbb{E}$: we already know how it works: objects are elements of the trees in $\mathbb{B}$ (via the map $E_T : \mathbb{B} \to \widehat{\mathbb{I}}$ that we did not make explicit). So objects are maps $e \to t$ where e is an elementary tree and t is a tree. In addition to those objects, there is one special object $\star$. For a fixed tree t, the fibre over t is simply the slice $\mathbb{I}/t$. Notice that if t has a ternary node, then there are 3! different objects in $\mathbb{E}$ lying over t. However, these 3! objects are linked by isomorphisms. The morphisms involving the special object $\star$ all go from $\star$ to some object of the form $e \to t$ where e is the trivial tree, included into t as a leaf or the root. And as before, there are two maps from $\star$ to the trivial (edge-marked) tree.

So once again, this monad T is polynomial, represented by

$$\begin{tikzcd}
& \mathbb{E} \arrow[r, "p"] \arrow[dl, "s"'] & \mathbb{B} \arrow[dr, "t"] & \\
\mathbb{I} & & T & \mathbb{I}
\end{tikzcd}$$

But in this case t is *not* a discrete fibration: in the fibre over an elementary tree n, we have all the trees with n leaves, and all the isomorphisms between them that fix the leaves. In the presence of nullary this may give non-trivial automorphisms. So the free-symmetric-multicategory monad is *not* a local right adjoint.

16.4 Local right adjoints

COPY MANY INTERESTING THINGS FROM MARK'S PAPERS [103] and notably [104].

Among the polynomial functors

$$\begin{tikzcd} \boldsymbol{PrSh}(I) \arrow[r, "s^*"] & \boldsymbol{PrSh}(E) \arrow[r, "p_*"] & \boldsymbol{PrSh}(B) \arrow[r, "t_!"] & \boldsymbol{PrSh}(J) \end{tikzcd}$$

a particularly useful and well-behaved class are the *local right adjoint* polynomial functors.

A *local right adjoint* is a functor $F : \mathscr{D} \to \mathscr{C}$ such that for every object $X \in \mathscr{D}$, the sliced functor

$$\begin{array}{rcl} \mathscr{D}/X & \longrightarrow & \mathscr{C}/FX \\ [D \to X] & \longmapsto & [FD \to FX] \end{array}$$

16.4.1 Proposition. *A polynomial functor is a local right adjoint if and only if t is a discrete fibration, I THINK.*

This is NOT true: let C be a category which is not Cauchy complete, and let $i : C \to \overline{C}$ be the inclusion into its Cauchy completion. Then $i_!$ is an equivalence, and in particular a local right adjoint. But it seems that i is not equivalent to a discrete fibration.

On a category with a terminal object, the condition for being a local right adjoint can be measured on 1 alone: the condition is that $\mathscr{D} \to \mathscr{C}/F1$ is a right adjoint.

16.4.2 Proposition. *Every local right adjoint functor between presheaf categories is presheaf polynomial (with t a discrete fibration).*

Proof. In the factorisation of $F : \boldsymbol{PrSh}(I) \to \boldsymbol{PrSh}(J)$ through $\boldsymbol{PrSh}(J)/F1 = \boldsymbol{PrSh}(\mathrm{el}(F1))$ we already found a discrete fibration $B := \mathrm{el}(F1) \to J$. This is the last leg of the bridge. Now we have a right adjoint $\boldsymbol{PrSh}(I) \to \boldsymbol{PrSh}(B)$. This is equivalent to having a left adjoint $\boldsymbol{PrSh}(B) \to \boldsymbol{PrSh}(I)$ and this is equivalent to having a functor $B \to \boldsymbol{PrSh}(I)$. This in turn is equivalent to having a distributor $I^{\mathrm{op}} \times B \to \boldsymbol{Set}$, and we find E as the apex of the associated mixed fibration. See 16.2.3 for details. □

16.4.3 Lemma. *Let $p : E \to B$ be a discrete fibration. Then $p_! : \widehat{E} \to \widehat{B}$ preserves and reflects all connected limits.*

Proof. Since p is a discrete fibration, the formula for $p_!$ (usually a coend formula) reduces to a sum:

$$(p_! X)(b) = \sum_{e \in E_b} X(e).$$

Now limits are computed pointwise, and sums commute with connected limits in ***Set***, hence connected limits are preserved. To see that they are also reflected, it is enough to observe that $\widehat{E}$ has all limits, and that $p_!$ is conservative. □

16.4.4 Lemma. *If $F : \widehat{D} \to \widehat{C}$ is a local right adjoint then F preserves all connected limits.*

Proof. Factor F as

$$\begin{tikzcd}
\widehat{D} \arrow[rr, "F"] \arrow[dr, "F_1"'] & & \widehat{C} \\
& \widehat{\mathrm{el}(F1)} \arrow[ur, "p_!"'] &
\end{tikzcd}$$

where $p : \mathrm{el}(F1) \to C$ is a discrete fibration. But F_1 is a right adjoint so it preserves all limits, and $p_!$ preserves connected limits by the previous lemma. □

16.4.5 Lemma. *If a category $\mathscr{C}$ has wide pullbacks and a terminal object, then it has all limits. (And if a functor out of $\mathscr{C}$ preserves wide pullbacks and the terminal object, then it preserves all limits.)*

Proof. Arbitrary limits can be constructed from products and equalisers, but products can be constructed as wide pullbacks over the terminal object, and equalisers can be constructed from pullbacks and products—the equaliser of $f, g : x \rightrightarrows y$ is the pullback

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d] & x \arrow[d, "{(x,g)}"] \\
x \arrow[r, "{(x,f)}"'] & x \times y.
\end{tikzcd}$$

□

16.4.6 Lemma. *If $F : \widehat{D} \to \widehat{C}$ preserves wide pullbacks, then it is a local right adjoint.*

Proof. Factor F as

$$\begin{tikzcd}
\widehat{D} \arrow[rr, "F"] \arrow[dr, "F_1"'] & & \widehat{C} \\
& \widehat{C}/F1 = \widehat{\mathrm{el}(F1)}. \arrow[ur, "p_!"'] &
\end{tikzcd}$$

By the first lemma $p_!$ reflects connected limits and in particular wide pullbacks. Therefore already F_1 preserves wide pullbacks. But since it obviously preserves the terminal object, by the previous lemma F_1 preserves all limits, and is therefore a right adjoint. □

Note. The characterisation of local right adjoints between presheaf categories as connected-limit preserving functors is quoted by Weber [104], Thm.2.13, but assuming F has a rank. He refers to Weber [103], but here the theorem is formulated and proved in terms of wide pullbacks (and still with a rank assumption). In the above sequence of lemmas I don't see where the rank assumption should come in. In [103] the proof is quite complicated and not limited to presheaf categories. . .

Chapter 17

[Generalised species and polynomial functors]

[35]

Chapter 18

Appendices

A Pullbacks

A pullback is characterised by the property that the fibres are isomorphic. . . Spell this out.

Some pullback formulae.

ck-rightcancel **A.1 Lemma.** *Given a diagram*

$$\begin{tikzcd}
\cdot \arrow[r] \arrow[d] & \cdot \arrow[d] \\
\cdot \arrow[r] \arrow[d] & \cdot \arrow[d] \\
\cdot \arrow[r] & \cdot
\end{tikzcd}$$

in which the big square and the bottom square are pullback squares, the top one is too.

Suppose $F \dashv G$, with $F : \mathscr{C} \to \mathscr{D}$ and $G : \mathscr{D} \to \mathscr{C}$. This means we have natural bijections $\mathrm{Hom}_{\mathscr{D}}(FX, Y) = \mathrm{Hom}_{\mathscr{C}}(X, GY)$. Suppose we have a cartesian square

$$\begin{tikzcd} FX' \arrow[r] \arrow[d, "F\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & Y' \arrow[d, "\beta"] \\ FX \arrow[r] & Y \end{tikzcd} \tag{18.1}$$

$\boxed{\texttt{sq1}}$

Then if the unit η is a cartesian natural transformation, then the corresponding square

$$\begin{tikzcd} X' \arrow[r] \arrow[d, "\alpha"'] \arrow[dr, phantom, "\lrcorner", very near start] & GY' \arrow[d, "G\beta"] \\ X \arrow[r] & GY \end{tikzcd} \tag{18.2}$$

$\boxed{\texttt{sq2}}$

is again cartesian. (This follows by first applying G to the square (since G is a right adjoint it preserves pullbacks), and then pre-pasting with the naturality square for η, which we assumed cartesian.)

Conversely, given a cartesian square like (18.2), if we assume that F preserves pullbacks and that the counit ε is a cartesian natural transformation, then the corresponding square (18.1) is again cartesian. (This follows by a similar argument.)

Bibliography

Abbott [1] Michael Abbott. *Categories of Containers*. PhD thesis, University of Leicester, 2003. Available from www.mcs.le.ac.uk/~ma139/docs/thesis.pdf.

t-al:fossacs03 [2] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. *Categories of containers*. In *Foundations of software science and computation structures*, vol. 2620 of Lecture Notes in Comput. Sci., pp. 23–38. Springer, Berlin, 2003.

h-Ghani:nested [3] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. *Representing nested inductive types using W-types*. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Automata, languages and programming*, vol. 3142 of Lecture Notes in Comput. Sci., pp. 59–71. Springer, Berlin, 2004.

ictly-positive [4] Michael Abbott, Thorsten Altenkirch, and Neil Ghani. *Containers: constructing strictly positive types*. Theoret. Comput. Sci. **342** (2005), 3–27.

t-et-al:tlca03 [5] Michael Abbott, Thorsten Altenkirch, Neil Ghani, and Conor McBride. *Derivatives of Containers*. In *Typed Lambda Calculi and Applications, TLCA*, 2003.

et-al:jpartial [6] Michael Abbott, Thorsten Altenkirch, Neil Ghani, and Conor McBride. *∂ for Data*. Fundamentae Informatica **65** (March 2005), 1 – 28. Special Issue on Typed Lambda Calculi and Applications 2003.

Adamek-Trnkova [7] Jiří Adámek and Věra Trnková. *Automata and algebras in categories*, vol. 37 of Mathematics and its Applications (East European Series). Kluwer Academic Publishers Group, Dordrecht, 1990.

rris-Altenkirch:lics09 [8] Thorsten Altenkirch and Peter Morris. *Indexed Containers*. In *Twenty-Fourth IEEE Symposium in Logic in Computer Science (LICS 2009)*, 2009. to appear.

Awodey:Cat [9] Steve Awodey. *Category theory*, vol. 49 of Oxford Logic Guides. The Clarendon Press Oxford University Press, New York, 2006.

Baez:counting [10] John Baez. The mysteries of counting: Euler characteristic versus homotopy cardinality. Talk given at the Conference in honour of Ross Street's sixtieth birthday: Categories in Algebra, Geometry and Mathematical Physics, Sydney, July 2005. Slides available from http://math.ucr.edu/home/baez/counting.

Baez-Dolan:9702 [11] John C. Baez and James Dolan. *Higher-dimensional algebra. III. n-categories and the algebra of opetopes*. Adv. Math. **135** (1998), 145–206. ArXiv:q-alg/9702014.

z-Dolan:finset-feynman [12] John C. Baez and James Dolan. *From finite sets to Feynman diagrams*. In B. Engquist and W. Schmid, editors, *Mathematics unlimited—2001 and beyond*, pp. 29–50. Springer-Verlag, Berlin, 2001. ArXiv:math.QA/0004133.

Barr-Wells [13] Michael Barr and Charles Wells. *Toposes, triples and theories*. No. 278 in Grundlehren der Mathematischen Wissenschaften. Springer-Verlag, 1985. Corrected reprint in Reprints in Theory and Applications of Categories, 12 (2005) (electronic).

Barr-Wells:computing [14] Michael Barr and Charles Wells. *Category theory for computing science*. Prentice Hall International Series in Computer Science. Prentice Hall International, New York, 1990.

Benabou:bicat [15] Jean Bénabou. *Introduction to bicategories*. In *Reports of the Midwest Category Seminar*, no. 47 in Lecture Notes in Mathematics, pp. 1–77. Springer-Verlag, Berlin, 1967.

Benabou:distributors [16] Jean Bénabou. Distributors at work, 2000. notes by Thomas Streicher.

Berger:Adv [17] Clemens Berger. *A cellular nerve for higher categories*. Adv. Math. **169** (2002), 118–175.

-Moerdijk:0512 [18] Clemens Berger and Ieke Moerdijk. *Resolution of coloured operads and rectification of homotopy algebras*. In *Categories in algebra, geometry and mathematical physics*, vol. 431 of Contemp. Math., pp. 31–58. Amer. Math. Soc., Providence, RI, 2007. ArXiv:math/0512576.

Labelle-Leroux [19] François Bergeron, Gilbert Labelle, and Pierre Leroux. *Combinatorial species and tree-like structures*, vol. 67 of Encyclopedia of Mathematics and its Applications. Cambridge University Press, Cambridge, 1998. Translated from the 1994 French original by Margaret Readdy, With a foreword by Gian-Carlo Rota.

al:Dyer-Lashof [20] Terrence Bisson and André Joyal. *The Dyer-Lashof algebra in bordism (extended abstract)*. C. R. Math. Rep. Acad. Sci. Canada **17** (1995), 135–140.

Borceux1 [21] Francis Borceux. *Handbook of categorical algebra. 1: Basic category theory*. Cambridge University Press, Cambridge, 1994.

Brown-Mosa [22] Ronald Brown and Ghafar H. Mosa. *Double categories, 2-categories, thin structures and connections*. Theory Appl. Categ. **5** (1999), No. 7, 163–175 (electronic).

Brun:0304495 [23] Morten Brun. *Witt vectors and Tambara functors*. Adv. Math. **193** (2005), 233–256. ArXiv:math/0304495.

Burroni:1971 [24] Albert Burroni. *T-catégories (catégories dans un triple)*. Cahiers Topologie Géom. Différentielle **12** (1971), 215–321.

boni-Johnstone [25] Aurelio Carboni and Peter Johnstone. *Connected limits, familial representability and Artin glueing*. Math. Structures Comput. Sci. **5** (1995), 441–459. With Corrigenda in Math. Structures Comput. Sci. **14** (2004), 185–187.

i-Lack-Walters [26] Aurelio Carboni, Stephen Lack, and Robert F. C. Walters. *Introduction to extensive and distributive categories*. J. Pure Appl. Algebra **84** (1993), 145–158.

Cockett:1990 [27] J. Robin B. Cockett. *List-arithmetic distributive categories: locoi*. J. Pure Appl. Algebra **66** (1990), 1–29.

Cohn:Universal [28] Paul Moritz Cohn. *Universal algebra*. Harper & Row Publishers, New York, 1965.

Conduche' [29] François Conduché. *Au sujet de l'existence d'adjoints à droite aux foncteurs "image réciproque" dans la catégorie des catégories*. C. R. Acad. Sci. Paris Sér. A-B **275** (1972), A891–A894.

etter:categorification [30] Louis Crane and David N. Yetter. *Examples of categorification*. Cah. Topol. Géom. Différ. Catég. **39** (1998), 3–25.

Dedekind:dieZahlen [31] Richard Dedekind. *Was sind und was sollen die Zahlen?* Braunschweig, 1888.

Diers:thesis [32] Yves Diers. *Catégories localisables*. PhD thesis, Université de Paris VI, 1977.

cLane:group-extensions [33] Samuel Eilenberg and Saunders MacLane. *Group extensions and homology*. Ann. of Math. (2) **43** (1942), 757–831.

a-GraciaBondia:0408145 [34] Héctor Figueroa and José M. Gracia-Bondía. *Combinatorial Hopf algebras in quantum field theory. I*. Rev. Math. Phys. **17** (2005), 881–976. ArXiv:hep-th/0408145.

ino-Hyland-Winskel:Esp [35] Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glen Winskel. *The Cartesian closed bicategory of generalised species of structures*. J. Lond. Math. Soc. (2) **77** (2008), 203–220.

Fiore:0608760 [36] Thomas M. Fiore. *Pseudo algebras and pseudo double categories*. J. Homotopy Relat. Struct. **2** (2007), 119–170. ArXiv:math/0608760.

Freyd:1972 [37] Peter Freyd. *Aspect of topoi*. Bull. Austral. Math. Soc. **7** (1972), 1–76.

Gambino-Hyland [38] Nicola Gambino and Martin Hyland. *Wellfounded Trees and Dependent Polynomial Functors*. In S. Berardi, M. Coppo, and F. Damiani, editors, *TYPES 2003*, vol. 3085 of Lecture Notes in Computer Science, pp. 210–225. Springer Verlag, Heidelberg, 2004.

Gambino-Kock:0906.4931 [39] Nicola Gambino and Joachim Kock. *Polynomial functors and polynomial monads*. Preprint, arXiv:0906.4931.

Gentzen:consistency [40] Gerhard Gentzen. *Die Widerspruchfreiheit der reinen Zahlentheorie*. Math. Ann. **112** (1936), 493–565.

zburg-Kapranov [41] Victor Ginzburg and Mikhail Kapranov. *Koszul duality for operads*. Duke Math. J. **76** (1994), 203–272. ArXiv:0709.1228.

Girard:1971 [42] Jean-Yves Girard. *Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types*. In *Proceedings of the Second Scandinavian Logic Symposium (Univ. Oslo, Oslo, 1970)*, pp. 63–92. Studies in Logic and the Foundations of Mathematics, Vol. 63. North-Holland, Amsterdam, 1971.

Girard:1988 [43] Jean-Yves Girard. *Normal functors, power series and λ-calculus*. Ann. Pure Appl. Logic **37** (1988), 129–177.

Giraud:mem [44] Jean Giraud. *Méthode de la descente*. Bull. Soc. Math. France Mém. **2** (1964). Available from http://www.numdam.org.

-Pare:adjoints [45] Marco Grandis and Robert Paré. *Adjoint for double categories*. Cah. Topol. Géom. Différ. Catég. **45** (2004), 193–240.

SGA1 [46] Alexander Grothendieck. *SGA 1: Revêtements Étales et Groupe Fondamental*. No. 224 in Lecture Notes in Mathematics. Springer-Verlag, Berlin, 1971. Available electronically as math.AG/0206203.

ock-Setzer:OLG [47] Peter Hancock and Anton Setzer. *Interactive programs and weakly final coalgebras in dependent type theory*. In *From sets and types to topology and analysis*, vol. 48 of Oxford Logic Guides, pp. 115–136. Oxford Univ. Press, Oxford, 2005.

mida:repr-mult [48] Claudio Hermida. *Representable multicategories*. Adv. Math. **151** (2000), 164–225.

Power:univ-alg [49] Martin Hyland and John Power. *The category theoretic understanding of universal algebra: Lawvere theories and monads*. In *Computation, meaning, and logic: articles dedicated to Gordon Plotkin*, vol. 172 of Electron. Notes Theor. Comput. Sci., pp. 437–458. Elsevier, Amsterdam, 2007.

yvernat:thesis [50] Pierre Hyvernat. *A logical investigation of interaction systems*. PhD thesis, Chalmers University, 2005.

Jay:1995 [51] C. Barry Jay. *A semantics for shape*. Sci. Comput. Programming **25** (1995), 251–283. ESOP '94 (Edinburgh, 1994).

Jay-Cockett [52] C. Barry Jay and J. Robin B. Cockett. *Shapely types and shape polymorphism*. In *Programming languages and systems—ESOP '94 (Edinburgh, 1994)*, vol. 788 of Lecture Notes in Comput. Sci., pp. 302–316. Springer, Berlin, 1994.

stone:cartesian-monads [53] Peter Johnstone. *Cartesian monads on toposes*. J. Pure Appl. Algebra **116** (1997), 199–220. Special volume on the occasion of the 60th birthday of Professor Peter J. Freyd.

Joyal:1981 [54] André Joyal. *Une théorie combinatoire des séries formelles*. Adv. Math. **42** (1981), 1–82.

:foncteurs-analytiques [55] André Joyal. *Foncteurs analytiques et espèces de structures*. In *Combinatoire énumérative (Montréal/Québec, 1985)*, vol. 1234 of Lecture Notes in Mathematics, pp. 126–159. Springer, Berlin, 1986.

Kelly:unified [56] G. Max Kelly. *A unified treatment of transfinite constructions for free algebras, free monoids, colimits, associated sheaves, and so on*. Bull. Austral. Math. Soc. **22** (1980), 1–83.

Kelly:basic-enriched [57] G. Max Kelly. *Basic concepts of enriched category theory*, vol. 64 of London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, 1982. Reprinted in *Repr. Theory Appl. Categ.* 10 (2005), vi+137 pp. (electronic).

Kelly:clubs92 [58] G. Max Kelly. *On clubs and data-type constructors*. In *Applications of categories in computer science (Durham, 1991)*, vol. 177 of London Math. Soc. Lecture Note Ser., pp. 163–190. Cambridge Univ. Press, Cambridge, 1992.

Kelly:operads [59] G. Max Kelly. *On the operads of J. P. May*. Repr. Theory Appl. Categ. (2005), 1–13 (electronic). Written 1972.

Kelly-Street:2cat [60] G. Max Kelly and Ross Street. *Review of the elements of* 2-*categories*. In *Category Seminar (Proc. Sem., Sydney, 1972/1973)*, no. 420 in Lecture Notes in Mathematics, pp. 75–103. Springer, Berlin, 1974.

Knuth:9207 [61] Donald E. Knuth. *Convolution polynomials*. Mathematica Journal **2** (1992), 67–78. ArXiv:math/9207221.

ock:strong1972 [62] Anders Kock. *Strong functors and monoidal monads*. Arch. Math. (Basel) **23** (1972), 113–120.

Kock:0807 [63] Joachim Kock. *Polynomial functors and trees*. Preprint, arXiv:0807.2874.

zoom [64] Joachim Kock, André Joyal, Michael Batanin, and Jean-François Mascari. *Polynomial functors and opetopes*. Preprint, arXiv:0706.1033.

Soibelman:0001 [65] Maxim Kontsevich and Yan Soibelman. *Deformations of algebras over operads and the Deligne conjecture*. In *Conférence Moshé Flato 1999, Vol. I (Dijon)*, vol. 21 of Math. Phys. Stud., pp. 255–307. Kluwer Acad. Publ., Dordrecht, 2000. ArXiv:math.QA/0001151.

rmal-monads-II [66] Stephen Lack and Ross Street. *The formal theory of monads. II*. J. Pure Appl. Algebra **175** (2002), 243–265. Special volume celebrating the 70th birthday of Professor Max Kelly.

amarche:thesis [67] François Lamarche. *Modelling polymorphism with categories*. PhD thesis, McGill University, 1988.

ambek:fixpoint [68] Joachim Lambek. *A fixpoint theorem for complete categories*. Math. Z. **103** (1968), 151–161.

ek:deductiveII [69] Joachim Lambek. *Deductive systems and categories. II. Standard constructions and closed categories*. In *Category Theory, Homology Theory and their Applications, I (Battelle Institute Conference, Seattle, Wash., 1968, Vol. One)*, pp. 76–122. Springer, Berlin, 1969.

Lang [70] Serge Lang. *Algebra*. Addison-Wesley, Reading, Mass., 1971.

Lawvere:thesis [71] F. William Lawvere. *Functorial semantics of algebraic theories and some algebraic problems in the context of functorial semantics of algebraic theories*. Repr. Theory Appl. Categ. (2004), 1–121 (electronic). Reprinted from Proc. Nat. Acad. Sci. U.S.A. **50** (1963), 869–872 and *Reports of the Midwest Category Seminar. II*, 41–61, Springer, Berlin, 1968.

Lawvere-Rosebrugh [72] F. William Lawvere and Robert Rosebrugh. *Sets for mathematics*. Cambridge University Press, Cambridge, 2003.

Lazard:analyseurs [73] Michel Lazard. *Lois de groupes et analyseurs*. Ann. Sci. Ecole Norm. Sup. (3) **72** (1955), 299–400.

Leinster:ten [74] Tom Leinster. *A survey of definitions of n-category*. Theory Appl. Categ. **10** (2002), 1–70 (electronic). ArXiv:math.CT/0107188.

Leinster:0305049 [75] Tom Leinster. *Higher Operads, Higher Categories*. London Math. Soc. Lecture Note Series. Cambridge University Press, Cambridge, 2004. ArXiv:math.CT/0305049.

MacLane:categories [76] Saunders Mac Lane. *Categories for the working mathematician, second edition*. No. 5 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1998.

Macdonald [77] Ian G. Macdonald. *Symmetric functions and Hall polynomials*. The Clarendon Press Oxford University Press, New York, 1979. Oxford Mathematical Monographs.

Manes [78] Ernest G. Manes. *Algebraic theories*. No. 26 in Graduate Texts in Mathematics. Springer-Verlag, New York, 1976.

Manes-Arbib:1986 [79] Ernest G. Manes and Michael A. Arbib. *Algebraic approaches to program semantics*. Texts and Monographs in Computer Science. Springer-Verlag, New York, 1986. AKM Series in Theoretical Computer Science.

Martins-Ferreira:0604 [80] Nelson Martins-Ferreira. *Pseudo-categories*. J. Homotopy Relat. Struct. **1** (2006), 47–78 (electronic). ArXiv:math/0604549.

k-Palmgren:Wellfounded [81] Ieke Moerdijk and Erik Palmgren. *Wellfounded trees in categories*. Annals of Pure and Applied Logic **104** (2000), 189–218.

Moerdijk-Weiss:0701295 [82] Ieke Moerdijk and Ittay Weiss. *On inner Kan complexes in the category of dendroidal sets*. Preprint, arXiv:math/0701295, To appear in Adv. Math.

Moerdijk-Weiss:0701293 [83] Ieke Moerdijk and Ittay Weiss. *Dendroidal sets*. Alg. Geom. Top. (2007), 1441–1470. ArXiv:math/0701293.

Moggi [84] Eugenio Moggi. *Notions of computation and monads*. Inform. and Comput. **93** (1991), 55–92. Selections from the 1989 IEEE Symposium on Logic in Computer Science.

oggi-Belle-Jay [85] Eugenio Moggi, Gianna Bellè, and C. Barry Jay. *Monads, shapely functors and traversals*. In *CTCS '99: Conference on Category Theory and Computer Science (Edinburgh)*, vol. 29 of Electron. Notes Theor. Comput. Sci., pp. Paper No. 29017, 22 pp. (electronic). Elsevier, Amsterdam, 1999.

th:programming [86] Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf type theory: an introduction*, vol. 7 of International Series of Monographs on Computer Science. The Clarendon Press Oxford University Press, New York, 1990.

son-Smith:MLTT [87] Bengt Nordström, Kent Petersson, and Jan M. Smith. *Martin-Löf's type theory*. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of logic in computer science, Vol. 5*, pp. 1–37. Oxford University Press, New York, 2000.

ano:arithmetic [88] Giuseppe Peano. *Arithmetices principia, nova methodo exposita*. Augustae Taurinorum, Ed. Fratres Bocca, 1889 (1889).

etersson-Synek [89] Kent Petersson and Dan Synek. *A set constructor for inductive sets in Martin-Löf type theory*. In D. E. Rydeheard, P. Dybjer, A. M. Pitts, and A. Poigné, editors, *Category Theoy and Computer Science (Manchester 1989)*, vol. 389 of Lecture Notes in Computer Science, pp. 128–140. Springer, 1989.

ashvili:survey [90] Teimuraz Pirashvili. *Polynomial functors over finite fields (after Franjou, Friedlander, Henn, Lannes, Schwartz, Suslin)*. In *Séminaire Bourbaki, Vol. 1999/2000*, vol. 276 of Astérisque, pp. 369–388. Société Mathématique de France, 2002.

Propp:0203 [91] James Propp. *Euler measure as generalized cardinality*. Preprint, arXiv:math.CO/0203289.

Propp:0204 [92] James Propp. *Exponentiation and Euler measure*. Algebra Universalis **49** (2003), 459–471. ArXiv:math.CO/0204009, Dedicated to the memory of Gian-Carlo Rota.

Reyes-Reyes-Zolfaghari [93] Marie La Palme Reyes, Gonzalo E. Reyes, and Houman Zolfaghari. *Generic figures and their glueings — a constructive approach to functor categories*. Polimetrica, 2004.

Reynolds:1974 [94] John C. Reynolds. *Towards a theory of type structure*. In *Programming Symposium (Proc. Colloq. Programmation, Paris, 1974)*, pp. 408–425. Lecture Notes in Comput. Sci., Vol. 19. Springer, Berlin, 1974.

Schanuel:negative [95] Stephen H. Schanuel. *Negative sets have Euler characteristic and dimension*. In *Category theory (Como, 1990)*, no. 1488 in Lecture Notes in Mathematics, pp. 379–385. Springer, Berlin, 1991.

Seely:lccc [96] Robert A. G. Seely. *Locally cartesian closed categories and type theory*. Mathematical Proceedings of the Cambridge Philosophical Society **95** (1984), 33–48.

Shulman:0706 [97] Michael Shulman. *Framed bicategories and monoidal fibrations*. Theory Appl. Categ. **20** (2008), 650–738 (electronic). ArXiv:0706.1286.

Spencer:1977 [98] Christopher B. Spencer. *An abstract setting for homotopy pushouts and pullbacks*. Cahiers Topologie Géom. Différentielle **18** (1977), 409–429.

Strachey:1967 [99] Christopher Strachey. Fundamental concepts in programming languages. Lecture notes for the International Summer School in Computer Programming, Copenhagen, August 1967.

Street:formal-monads [100] Ross Street. *The formal theory of monads*. J. Pure Appl. Algebra **2** (1972), 149–168.

Tambara:CommAlg [101] Daisuke Tambara. *On multiplicative transfer*. Comm. Algebra **21** (1993), 1393–1420.

or:quantitativedomains [102] Paul Taylor. *Quantitative domains, groupoids and linear logic*. In *Category theory and computer science (Manchester, 1989)*, vol. 389 of Lecture Notes in Comput. Sci., pp. 155–181. Springer, Berlin, 1989. See also http://www.paultaylor.eu/stable/.

Weber:TAC13 [103] Mark Weber. *Generic morphisms, parametric representations and weakly Cartesian monads*. Theory Appl. Categ. **13** (2004), 191–234 (electronic).

Weber:TAC18 [104] Mark Weber. *Familial 2-functors and parametric right adjoints*. Theory Appl. Categ. **18** (2007), 665–732 (electronic).

ood:proarrowII [105] Richard J. Wood. *Proarrows. II*. Cahiers Topologie Géom. Différentielle Catég. **26** (1985), 135–168.

lastpage