Dependent Inductive and Coinductive Types are Fibrational Dialgebras

Henning Basold
Radboud University, iCIS, Intelligent Systems
CWI, Amsterdam, The Netherlands
h.basold@cs.ru.nl

In this paper, I establish the categorical structure necessary to interpret dependent inductive and coinductive types. It is well-known that dependent type theories à la Martin-Löf can be interpreted using fibrations. Modern theorem provers, however, are based on more sophisticated type systems that allow the definition of powerful inductive dependent types (known as inductive families) and, somewhat limited, coinductive dependent types. I define a class of functors on fibrations and show how data type definitions correspond to initial and final dialgebras for these functors. This description is also a proposal of how coinductive types should be treated in type theories, as they appear here simply as dual of inductive types. Finally, I show how dependent data types correspond to algebras and coalgebras, and give the correspondence to dependent polynomial functors.

1 Introduction

It is a well-established fact that the semantics of inductive data types without term dependencies can be given by initial algebras, whereas the semantics of coinductive types can be given by final coalgebras. However, for types that depend on terms, the situation is not as clear-cut.

Partial answers for inductive types can be found in [3, 8, 9, 11, 14, 19, 20], where semantics have been given for inductive types through polynomial functors in the category of set families or in locally Cartesian closed categories. Similarly, semantics for non-dependent coinductive types have been given in [1, 2, 6] by using polynomial functors on locally Cartesian closed categories. Finally, an interpretation for Martin-Löf type theory (without recursive type definitions) has been given in [21] and corrected in [16].

So far, we are, however, lacking a full picture of dependent coinductive types that arise as duals of dependent inductive types. To actually get such a picture, I extend in the present work Hagino's idea [13], of using dialgebras to describe data types, to dependent types. This emphasises the actual structure behind (co)inductive types as their are used in systems like Agda.[1] Moreover, dialgebras allow for a direct interpretation of types in this categorical setup, without going through translations into, for example, polynomial functors.

Having defined the structures we need to interpret dependent data types, it is natural to ask whether this structure is actually sensible. The idea, pursued here, is that we want to obtain initial and final dialgebras from initial algebras and final coalgebras for polynomial functors. This is achieved by showing that the dialgebras in this work correspond to algebras and coalgebras, and that their fixed points can be constructed from fixed points of polynomial functors (in the sense of [12]).

[1] It should be noted that, for example, Coq treats coinductive types differently. In fact, the route taken in Agda with copatterns and in this work is much better behaved.

R. Matthes, M. Mio (Eds.): Fixed Points
in Computer Science 2015 (FICS 2015)
EPTCS 191, 2015, pp. 3–17, doi:10.4204/EPTCS.191.3

To summarise, this paper makes the following contributions. First, we get a precise description of the categorical structure necessary to interpret inductive and coinductive data types, which can be seen as categorical semantics for an extension of the inductive and (copattern-based) coinductive types of Agda. The second contribution is a reduction to fixed points of polynomial functors.

What has been left out, because of space constraints, is an analysis of the structures needed to obtain induction and coinduction principles. Moreover, to be able to get a sound interpretation, with respect to type equality of dependent types, we need to require a Beck-Chevalley condition. This condition can be formulated for general (co)inductive types, but is also not given here.

Related work As already mentioned, there is an enormous body of work on obtaining semantics for (dependent) inductive, and to some extent, coinductive types, see [3, 11, 14, 20]. In the present work, we will mostly draw from [2] and [12]. Categorical semantics for basic Martin-Löf type theory have been developed, for example, in [16]. An interpretation, closer to the present work, is given in terms of fibrations by Jacobs [17]. In the first part of the paper, we develop everything on rather arbitrary fibrations, which makes the involved structure more apparent. Only in the second part, where we reduce data types to polynomial functors, we will work with slice categories, since most of the work on polynomial functors in that setting [2, 12]. Last, but not least, the starting idea of this paper is of course inspired by the dialgebras of Hagino [13]. These have also been applied to give semantics to induction-induction [4] schemes.

Outline The rest of the paper is structured as follows. In Section 2, we analyse a typical example of a dependent inductive type, namely vectors, that is, lists indexed by their length. We develop from this example a description of inductive and coinductive dependent data types in terms of dialgebras in fibrations. This leads to the requirements on a fibration, given in Section 3, that allow the interpretation of data types. In the same section, we show how dependent and fibrewise (co)products arise canonically in such a structure, and we give an example of a coinductive type (partial streams) that can only be treated in Agda through a cumbersome encoding. The reduction of dependent data types to polynomial functors is carried out in Section 4, and finish with concluding remarks in Section 5.

Acknowledgement I would like to thank the anonymous reviewers, who gave very valuable feedback and pointed me to some more literature.

2 Fibrations and Dependent Data Types

In this section we introduce *dependent data types* as initial and final dialgebras of certain functors on fibres of fibrations. We go through this setup step by step.

Let us start with dialgebras and their homomorphisms.

Definition 2.1. Let $\mathbf{C}$ and $\mathbf{D}$ be categories and $F, G : \mathbf{C} \to \mathbf{D}$ functors. An (F,G)-*dialgebra* is a morphism $c : FA \to GA$ in $\mathbf{D}$, where A is an object in $\mathbf{C}$. Given dialgebras $c : FA \to GA$ and $d : FB \to GB$, a morphism $h : A \to B$ is said to be a (dialgebra) *homomorphism* from c to d, if $Gh \circ c = d \circ Fh$. This allows us to form a category $\mathrm{DiAlg}\,(F,G)$, in which objects are pairs (A,c) with $A \in \mathbf{C}$ and $c : FA \to GA$, and morphisms are dialgebra homomorphisms.

The following example shows that dialgebras arise naturally from data types.

Example 2.2. Let A be a set, we denote by A^n the n-fold product of A, that is, lists of length n. Vectors over A are given by the set family $\mathrm{Vec}A = \{A^n\}_{n\in\mathbb{N}}$, which is an object in the category $\mathbf{Set}^{\mathbb{N}}$ of families

indexed by $\mathbb{N}$. In general, this category is given for a set I by

$$\mathbf{Set}^I = \begin{cases} \text{objects} & X = \{X_i\}_{i\in I} \\ \text{morphisms} & f = \{f_i : X_i \to Y_i\}_{i\in I} \end{cases}.$$

Vectors come with two constructors: $\text{nil} : \mathbf{1} \to A^0$ for the empty vector and prefixing $\text{cons}_n : A \times A^n \to A^{n+1}$ of vectors with elements of A. We note that $\text{nil} : \{\mathbf{1}\} \to \{A^0\}$ is a morphism in the category $\mathbf{Set}^{\mathbf{1}}$ of families indexed by the one-element set $\mathbf{1}$, whereas $\text{cons} = \{\text{cons}_n\} : \{A \times A^n\}_{n\in\mathbb{N}} \to \{A^{n+1}\}_{n\in\mathbb{N}}$ is a morphism in $\mathbf{Set}^{\mathbb{N}}$.

Let $F, G : \mathbf{Set}^{\mathbb{N}} \to \mathbf{Set}^{\mathbf{1}} \times \mathbf{Set}^{\mathbb{N}}$ be the functors into the product of $\mathbf{Set}^{\mathbf{1}}$ and $\mathbf{Set}^{\mathbb{N}}$ with

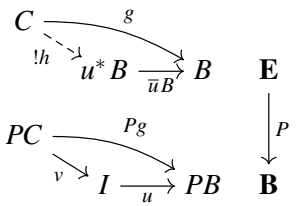
$$F(X) = (\{\mathbf{1}\}, \{A \times X_n\}_{n\in\mathbb{N}}) \qquad G(X) = (\{X_0\}, \{X_{n+1}\}_{n\in\mathbb{N}}).$$

Using these, we find that $(\text{nil}, \text{cons}) : F(\text{Vec}A) \to G(\text{Vec}A)$ is an (F,G)-dialgebra, in fact, it is the *initial* (F,G)-dialgebra.

Definition 2.3. An (F,G)-dialgebra $c : FA \to GA$ is called *initial*, if for every (F,G)-dialgebra $d : FB \to GB$ there is a unique homomorphism h from c to d, the *inductive extension* of d. Dually, (A,c) is *final*, provided there is a unique homomorphism h from any other dialgebra (B,d) into c. Here, h is the *coinductive extension* of d.

Having found the algebraic structure underlying vectors, we continue by exploring how we can handle the change of indices in the constructors. It turns out that this is most conveniently done by using fibrations.

Definition 2.4. Let $P : \mathbf{E} \to \mathbf{B}$ be a functor, where the $\mathbf{E}$ is called the *total* category and $\mathbf{B}$ the *base* category. A morphism $f : A \to B$ in $\mathbf{E}$ is said to be *cartesian over* $u : I \to J$, provided that i) $Pf = u$, and ii) for all $g : C \to B$ in $\mathbf{E}$ and $v : PC \to I$ with $Pg = u \circ v$ there is a unique $h : C \to A$ such that $f \circ h = g$. For P to be a *fibration*, we require that for every $B \in \mathbf{E}$ and $u : I \to PB$ in $\mathbf{B}$, there is a cartesian morphism $f : A \to B$ over u. Finally, a fibration is *cloven*, if it comes with a unique choice for A and f, in which case we denote A by $u^* B$ and f by $\overline{u}B$, as displayed in the diagram on the right.



At first sight, this definition is arguably intimidating to someone who has never been exposed to fibrations. The idea is that the base category $\mathbf{B}$ contains as objects the indices of objects in $\mathbf{E}$, and as morphisms substitutions. The result of carrying out a substitution on indices, is captured by the Cartesian lifting property. Let us illustrate this on set families. We define $\text{Fam}(\mathbf{Set})$ to be the category

$$\text{Fam}(\mathbf{Set}) = \begin{cases} \text{objects} & (I, X : I \to \mathbf{Set}),\ I \text{ a set} \\ \text{morphisms} & (u,f) : (I,X) \to (J,Y) \text{ with } u : I \to J \text{ and } \{f_i : X_i \to Y_{u(i)}\}_{i\in I} \end{cases}$$

in which composition is defined by

$$(v,g) \circ (u,f) = \left(v \circ u, \{X_i \xrightarrow{f_i} Y_{u(i)} \xrightarrow{g_{u(i)}} Z_{v(u(i))}\}_{i\in I}\right).$$

A concrete object is the pair $(\mathbb{N}, \text{Vec}A)$, where $\text{Vec}A$ is the family of vectors from Ex. 2.2.

We define a cloven fibration on set families. Let $P : \text{Fam}(\mathbf{Set}) \to \mathbf{Set}$ be the projection on the first component, that is, $P(I,X) = I$ and $P(u,f) = u$. For a family (J,Y) and a function $u : I \to J$, we define

$u^*Y = \{Y_{u(i)}\}_{i\in I}$ and $\overline{u}Y = \big(u, \{\mathrm{id} : Y_{u(i)} \to Y_{u(i)}\}_{i\in I}\big)$. Then, for each $(w,g) : (K,Z) \to (J,Y)$ and $v : K \to I$ with $w = u \circ v$, we can define the morphism $(K,Z) \to (I,u^*Y)$ to be (v,h) with $h_k : Z_k \to Y_{u(v(k))}$ and $h_k = g_k$, since $u(v(k)) = w(k)$.

An important concept is the *fibre above* an object $I \in \mathbf{B}$, given by the category

$$\mathbf{P}_I = \begin{cases} \text{objects} & A \in \mathbf{E} \text{ with } P(A) = I \\ \text{morphisms} & f : A \to B \text{ with } P(f) = \mathrm{id}_I \end{cases}.$$

In a cloven fibration, we can use the Cartesian lifting to define for each $u : I \to J$ in $\mathbf{B}$ a functor $u^* : \mathbf{P}_J \to \mathbf{P}_I$, together with natural isomorphisms $\mathrm{Id}_{\mathbf{P}_I} \cong \mathrm{id}_I^*$ and $u^* \circ v^* \cong (v \circ u)^*$, see [17, Sec. 1.4]. The functor u^* is called *reindexing* along u.

Assumption 2.5. We assume all fibrations to be cloven in this work.

We are now in the position to take a more abstract look at our initial example.

Example 2.6. First, we note that the fibre of $\mathrm{Fam}(\mathbf{Set})$ above I is isomorphic to $\mathbf{Set}^I$. Let then $z : \mathbf{1} \to \mathbb{N}$ and $s : \mathbb{N} \to \mathbb{N}$ be $z(*) = 0$ and $s(n) = n+1$, giving us reindexing functors $z^* : \mathbf{Set}^{\mathbb{N}} \to \mathbf{Set}^{\mathbf{1}}$ and $s^* : \mathbf{Set}^{\mathbb{N}} \to \mathbf{Set}^{\mathbb{N}}$. By their definition, $z^*(X) = \{X_0\}$ and $s^*(X) = \{X_{n+1}\}_{n\in\mathbb{N}}$, hence the functor G, we used to describe vectors as dialgebra, is $G = \langle z^*, s^*\rangle$. In Sec. 3, we address the structure of F.

We generalise this situation to account for arbitrary data types.

Definition 2.7. Let $P : \mathbf{E} \to \mathbf{B}$ be a fibration. A *(dependent) data type signature*, parameterised by a category $\mathbf{C}$, is a pair (F,u) consisting of

- a functor $F : \mathbf{C} \times \mathbf{P}_I \to \mathbf{D}$ with $\mathbf{D} = \prod_{k=1}^n \mathbf{P}_{J_k}$ for some $n \in \mathbb{N}$ and $J_k, I \in \mathbf{B}$, and
- a family u of n morphisms in $\mathbf{B}$ with $u_k : J_k \to I$ for $k = 1,\dots,n$.

A family u as above induces a functor $\langle u_1^*,\dots,u_n^*\rangle : \mathbf{P}_I \to \mathbf{D}$, which we will often denote by G_u. This will enable us to define data types for such signatures, but let us first look at an example for the case $\mathbf{C} = \mathbf{1}$, that is, if $F : \mathbf{P}_I \to \mathbf{D}$ is not parameterised.

Example 2.8. A fibration $P : \mathbf{E} \to \mathbf{B}$ is said to have dependent coproducts and products, if for each $f : I \to J$ in $\mathbf{B}$ there are functors $\coprod_f$ and $\prod_f$ from $\mathbf{P}_I$ to $\mathbf{P}_J$ that are respectively left and right adjoint to f^*. For each $X \in \mathbf{P}_I$, we can define a signature, such that $\coprod_f(X)$ and $\prod_f(X)$ arise as data types for these signatures, as follows. Define the constant functor

$$K_X : \mathbf{P}_J \to \mathbf{P}_I \qquad K_X(Y) = X \qquad K_X(g) = \mathrm{id}_X\,.$$

Then (K_X, f) is the signature for coproducts and products. For example, the unit η of the adjunction $\coprod_f \dashv f^*$ will be the initial (K_X, f^*)-dialgebra $\eta_X : K_X(\coprod_f(X)) \to f^*(\coprod_f(X))$, using that $K_X(\coprod_f(X)) = X$. We come back to this in Ex. 2.10. □

To define data types in general, we allow them to have additional parameters, that is, we allow signatures (F,u), where $F : \mathbf{C} \times \mathbf{P}_I \to \mathbf{D}$ and $\mathbf{C}$ is a non-trivial category. Let us first fix some notation. We put $F(V,-)(X) = F(V,X)$ for $V \in \mathbf{C}$, which is a functor $\mathbf{P}_I \to \mathbf{D}$. Assume that the initial $(F(V,-), G_u)$-dialgebra $\alpha_V : F(V,\Phi_V) \to G_u(\Phi_V)$ and final $(G_u, F(V,-))$-dialgebra $\xi_V : G_u(\Omega_V) \to F(V,\Omega_V)$ exist. Then we can define functors $\mu(\widehat{F},\widehat{G_u}) : \mathbf{C} \to \mathbf{P}_I$ and $\nu(\widehat{G_u},\widehat{F}) : \mathbf{C} \to \mathbf{P}_I$, analogous to [18], by

$$\mu(\widehat{F},\widehat{G_u})(V) = \Phi_V \qquad \mu(\widehat{F},\widehat{G_u})(f : V \to W) = \big(\alpha_W \circ F(f, \mathrm{id}_{\Phi_W})\big)^-$$
$$\nu(\widehat{G_u},\widehat{F})(V) = \Omega_V \qquad \nu(\widehat{G_u},\widehat{F})(f : V \to W) = \big(F(f, \mathrm{id}_{\Omega_V}) \circ \xi_V\big)^\sim,$$

where the bar and tilde superscripts denote the inductive and coinductive extensions, that is, the unique homomorphism given by initiality and finality, respectively. The reason for the notation $\mu(\widehat{F},\widehat{G_u})$ and $\nu(\widehat{G_u},\widehat{F})$ is that these are initial and final dialgebras for the functors

$$\widehat{F},\widehat{G_u}:[\mathbf{C},\mathbf{P}_I]\to[\mathbf{C},\mathbf{D}]\qquad \widehat{F}(H)=F\circ\langle \mathrm{Id}_{\mathbf{C}},H\rangle \qquad \widehat{G_u}(H)=G_u\circ H$$

on functor categories. That the families α_V and ξ_V are natural in V follows directly from the definition of the functorial action as (co)inductive extensions. Hence, they give rise to dialgebras $\alpha\colon \widehat{F}(\mu(\widehat{F},\widehat{G_u}))\Rightarrow \widehat{G_u}(\mu(\widehat{F},\widehat{G_u}))$ and $\xi\colon \widehat{G_u}(\nu(\widehat{G_u},\widehat{F}))\Rightarrow \widehat{F}(\nu(\widehat{G_u},\widehat{F}))$.

Definition 2.9. Let (F,u) be a data type signature. An *inductive data type* (IDT) for (F,u) is an initial $(\widehat{F},\widehat{G_u})$-dialgebra with carrier $\mu(\widehat{F},\widehat{G_u})$. Dually, a coinductive data type (CDT) for (F,u) is a final $(\widehat{G_u},\widehat{F})$-dialgebra, note the order, with the carrier being denoted by $\nu(\widehat{G_u},\widehat{F})$. If $\mathbf{C}=\mathbf{1}$, we drop the hats from the notation.

Example 2.10. We turn the definition of the product and coproduct from Ex. 2.8 into actual functors. The observation we use is that the projection functor $\pi_1:\mathbf{P}_I\times\mathbf{P}_J\to\mathbf{P}_I$ gives us a "parameterised" constant functor: $K^J_A=\pi_1(A,-)$. If we are given $f:I\to J$ in $\mathbf{B}$, then we use the signature (π_1,f), and define $\coprod_f=\mu(\widehat{\pi_1},\widehat{f^*})$ and $\prod_f=\nu(\widehat{f^*},\widehat{\pi_1})$. We check the details of this definition in Thm. 3.2.

3 Data Type Completeness

We now define a class of signatures and functors that should be seen as categorical language for, what is usually called, strictly positive types [3], positive generalised abstract data types [14] or descriptions [8, 9]. Note, however, that none of these treat coinductive types. A *non-dependent* version of strictly positive types that include coinductive types are given in [2].

Let us first introduce some notation. Given categories $\mathbf{C}_1$ and $\mathbf{C}_2$ and an object $A\in\mathbf{C}_1$, we denote by $K^{\mathbf{C}_1}_A:\mathbf{C}_1\to\mathbf{C}_2$ the functor mapping constantly to A. The projections on product categories are denoted, as usual, by $\pi_k:\mathbf{C}_1\times\mathbf{C}_2\to\mathbf{C}_k$. Using these notations, we can define what we understand to be a data type by mutual induction.

Definition 3.1. A fibration $P:\mathbf{E}\to\mathbf{B}$ is *data type complete*, if all IDTs and CDTs for *strictly positive signatures* $(F,u)\in\mathscr{S}$ exist, where $\mathscr{S}$ is given by the following rule.

$$\frac{\mathbf{D}=\prod_{i=1}^n\mathbf{P}_{J_i}\qquad F\in\mathscr{D}_{\mathbf{C}\times\mathbf{P}_I\to\mathbf{D}}\qquad u=(u_1:J_1\to I,\dots,u_n:J_n\to I)}{(F,u)\in\mathscr{S}_{\mathbf{C}\times\mathbf{P}_I\to\mathbf{D}}}$$

The functors in $\mathscr{D}$ are given by the following rules, assuming that P is data type complete.

$$\frac{A\in\mathbf{P}_J}{K^{\mathbf{P}_I}_A\in\mathscr{D}_{\mathbf{P}_I\to\mathbf{P}_J}}\qquad \frac{\mathbf{C}=\prod_{i=1}^n\mathbf{P}_{I_i}}{\pi_k\in\mathscr{D}_{\mathbf{C}\to\mathbf{P}_{I_k}}}\qquad \frac{f:J\to I\text{ in }\mathbf{B}}{f^*\in\mathscr{D}_{\mathbf{P}_I\to\mathbf{P}_J}}\qquad \frac{F_1\in\mathscr{D}_{\mathbf{P}_I\to\mathbf{P}_K}\qquad F_2\in\mathscr{D}_{\mathbf{P}_K\to\mathbf{P}_J}}{F_2\circ F_1\in\mathscr{D}_{\mathbf{P}_I\to\mathbf{P}_J}}$$

$$\frac{F_i\in\mathscr{D}_{\mathbf{P}_I\to\mathbf{P}_{J_i}}\qquad i=1,2}{\langle F_1,F_2\rangle\in\mathscr{D}_{\mathbf{P}_I\to\mathbf{P}_{J_1}\times\mathbf{P}_{J_2}}}\qquad \frac{(F,u)\in\mathscr{S}_{\mathbf{C}\times\mathbf{P}_I\to\mathbf{D}}}{\mu(\widehat{F},\widehat{G_u})\in\mathscr{D}_{\mathbf{C}\to\mathbf{P}_I}}\qquad \frac{(F,u)\in\mathscr{S}_{\mathbf{C}\times\mathbf{P}_I\to\mathbf{D}}}{\nu(\widehat{G_u},\widehat{F})\in\mathscr{D}_{\mathbf{C}\to\mathbf{P}_I}}$$

This mutual induction is well-defined, as it can be stratified in the nesting of fixed points.

As a first sanity check, we show that a data type complete fibration has, both, fibrewise and dependent (co)products. These are instances of the following, more general, result.

Theorem 3.2. *Suppose $P : \mathbf{E} \to \mathbf{B}$ is a data type complete fibration. Let $\mathbf{C} = \prod_{i=1}^{m} \mathbf{P}_{K_i}$ and $\pi_1 : \mathbf{C} \times \mathbf{P}_I \to \mathbf{C}$ be the first projection. If $G_u : \mathbf{P}_I \to \mathbf{C}$ is such that (π_1, u) is a signature, then we have the following adjoint situation:*

$$\mu(\widehat{\pi_1}, \widehat{G_u}) \dashv G_u \dashv \nu(\widehat{G_u}, \widehat{\pi_1}).$$

Proof. We only show how the adjoint transposes are obtained in the case of inductive types. Concretely, for a tuple $V \in \mathbf{C}$ and an object $A \in \mathbf{P}_I$, we need to prove the correspondence

$$\frac{f: \quad \mu(\widehat{\pi_1}, \widehat{G_u})(V) \longrightarrow A \qquad \text{in } \mathbf{P}_I}{g: \qquad\qquad V \longrightarrow G_u A \quad \text{in } \mathbf{C}}$$

Let us use the notation $H = \mu(\widehat{\pi_1}, \widehat{G_u})$, then the choice of π_1 implies that the initial $(\widehat{\pi_1}, \widehat{G_u})$-dialgebra is of type $\alpha\colon\ \mathrm{Id}_{\mathbf{C}} \Rightarrow G_u \circ H$, since $\widehat{\pi_1}(H) = \pi_1 \circ \langle \mathrm{Id}_{\mathbf{C}}, H \rangle = \mathrm{Id}_{\mathbf{C}}$ and $\widehat{G_u}(H) = G_u \circ H$. This allows us to use as transpose of f the morphism $V \xrightarrow{\alpha_V} G_u(H(V)) \xrightarrow{G_u f} G_u A$. As transpose of g, we use the inductive extension of $\widehat{\pi_1}(K_A^{\mathbf{C}})(V) = V \xrightarrow{g} G_u A = \widehat{G_u}(K_A^{\mathbf{C}})(V)$. The proof that this correspondence is natural and bijective follows straightforwardly from initiality. For coinductive types, the result is given by duality. □

This gives fibrewise coproducts by $+_I = \mu(\widehat{\pi_1}, \widehat{G_u})$ and products by $\times_I = \nu(\widehat{G_u}, \widehat{\pi_1})$, using $u = (\mathrm{id}_I, \mathrm{id}_I)$. Dependent (co)products along $f : I \to J$ use $u = f$, see Ex. 2.10.

There are many more examples of data types that exist in a data type complete fibration. We describe three fundamental ones.

Example 3.3.

1. The first example are initial and final objects inside the fibres $\mathbf{P}_I$. Since an initial object is characterised by having a unique morphism *to* every other object, we define it as an initial dialgebra, namely $\mathbf{0}_I = \mu(\mathrm{Id}, \mathrm{id}_I^*)$. Then there is, for each $A \in \mathbf{P}_I$, a unique morphism $!^A : \mathbf{0}_I \to A$ given as inductive extension of id_A. Dually, we define the terminal object $\mathbf{1}_I$ in $\mathbf{P}_I$ to be $\nu(\mathrm{id}_I^*, \mathrm{Id})$ and for each A the corresponding unique morphism $!_A : A \to \mathbf{1}_I$ as the coinductive extension of id_A.

   Note that this also follows from Thm. 3.2, if we require that (co)inductive data types also exist if $\mathbf{C} = \mathbf{1}$ (the empty product) and $u = \{\}$ (empty family of morphisms). This allows us to define the initial and final object as functors $\mathbf{1} \to \mathbf{P}_I$.

2. There are several definable notions of equality, provided that $\mathbf{B}$ has binary products. A generic one is propositional equality $\mathrm{Eq} : \mathbf{P}_I \to \mathbf{P}_{I \times I}$, the left adjoint to the contraction functor $\delta^* : \mathbf{P}_{I \times I} \to \mathbf{P}_I$, which is induced by the diagonal $\delta : I \to I \times I$. Thus it is given by the dependent coproduct $\mathrm{Eq} = \coprod_\delta$ and the constructor $\mathrm{refl}_X : X \to \delta^*(\mathrm{Eq} X)$.

3. Assume that there is an object A^ω in $\mathbf{B}$ of streams over A, together with projections to head and tail. Then we can define bisimilarity between streams as CDT for the signature

$$F, G_u : \mathbf{P}_{(A^\omega)^2} \to \mathbf{P}_{(A^\omega)^2} \times \mathbf{P}_{(A^\omega)^2}$$
$$F = \left\langle (\mathrm{hd} \times \mathrm{hd})^* \circ K_{\mathrm{Eq}(A)}, (\mathrm{tl} \times \mathrm{tl})^* \right\rangle \qquad \text{and} \qquad u = (\mathrm{id}_{A^\omega \times A^\omega}, \mathrm{id}_{A^\omega \times A^\omega}).$$

   Note that there is a category $\mathrm{Rel}(\mathbf{E})$ of binary relations in $\mathbf{E}$ by forming the pullback of P along $\Delta : \mathbf{B} \to \mathbf{B}$ with $\Delta(I) = I \times I$, see [15]. Then we can reinterpret F and G_u by

$$F, G_u : \mathrm{Rel}(\mathbf{E})_{A^\omega} \to \mathrm{Rel}(\mathbf{E})_{A^\omega} \times \mathrm{Rel}(\mathbf{E})_{A^\omega}$$
$$F = \langle \mathrm{hd}^\# \circ K_{\mathrm{Eq}(A)}, \mathrm{tl}^\# \rangle \qquad \text{and} \qquad G_u = \langle \mathrm{id}_{A^\omega}^\#, \mathrm{id}_{A^\omega}^\# \rangle,$$

where $(-)^{\#}$ is reindexing in $\mathrm{Rel}(\mathbf{E})$. The final (G_u, F)-dialgebra is a pair of morphisms

$$(\mathrm{hd}^{\sim}_A : \mathrm{Bisim}_A \to \mathrm{hd}^{\#}(\mathrm{Eq}(A)), \mathrm{tl}^{\sim}_A : \mathrm{Bisim}_A \to \mathrm{tl}^{\#}(\mathrm{Bisim}_A)).$$

Bisim_A should be thought of to consist of all bisimilarity proofs. Coinductive extensions yield the usual coinduction proof principle, allowing us to prove bisimilarity by establishing a bisimulation relation $R \in \mathrm{Rel}(\mathbf{E})_{A^\omega}$ together with $h : R \to \mathrm{hd}^{\#}(\mathrm{Eq}(A))$ and $t : R \to \mathrm{tl}^{\#}(R)$, saying that the heads of related streams are equal and that the tails of related streams are again related.

The last example, we give, shall illustrate the additional capabilities of CDTs in the present setup over those currently available in Agda. However, one should note that coinductive types in Agda provide extra power in the sense that destructors can refer to each other. This is equivalent to having a strong coproduct [17, Sec. 10.1 and Def. 10.5.2], which we do not require in the setup of this work and thus A proof of this equivalence is left out because of space constraints.

Example 3.4. A partial stream is a stream together with a, possibly infinite, depth up to which it is defined. Assume that there is an object $\mathbb{N}^\infty$ of natural numbers extended with infinity and a successor map $s_\infty : \mathbb{N}^\infty \to \mathbb{N}^\infty$ in $\mathbf{B}$, we will see how these can be defined below. Then partial streams correspond to the following type declaration.

codata PStr (A : **Set**) : $\mathbb{N}^\infty \to$ **Set where**
  hd : (n : $\mathbb{N}^\infty$) $\to$ PStr (s_∞ n) $\to$ A
  tl : (n : $\mathbb{N}^\infty$) $\to$ PStr (s_∞ n) $\to$ PStr n

In an explicit, set-theoretic notation, we can define them as a family indexed by $n \in \mathbb{N}^\infty$:

$$\mathrm{PStr}(A)_n = \{s : \mathbb{N} \rightharpoonup A \mid \forall k < n. k \in \mathrm{dom}\, s \wedge \forall k \geq n. k \notin \mathrm{dom}\, s\},$$

where the order on $\mathbb{N}^\infty$ is given by extending that of the natural numbers with ∞ as strict top element, i.e., such that $k < \infty$ for all $k \in \mathbb{N}$.

The interpretation of $\mathrm{PStr}(A)$ for $A \in \mathbf{P_1}$ in a data type complete fibration is given, similarly to vectors, as the carrier of the final (G_u, F)-dialgebra, where

$$G_u, F : \mathbf{P}_{\mathbb{N}^\infty} \to \mathbf{P}_{\mathbb{N}^\infty} \times \mathbf{P}_{\mathbb{N}^\infty} \qquad G_u = \langle s^*_\infty, s^*_\infty \rangle \qquad F = \left\langle K^{\mathbb{N}^\infty}_{\overline{A}}, \mathrm{Id} \right\rangle$$

and $\overline{A} = !^*_{\mathbb{N}^\infty}(A) \in \mathbf{P}_{\mathbb{N}^\infty}$ is the weakening of A using $!_{\mathbb{N}^\infty} : \mathbb{N}^\infty \to \mathbf{1}$. The idea of this signature is that the head and tail of partial streams are defined only on those partial streams that are defined in, at least, the first position. On set families, partial streams are given by the dialgebra $\xi = (\mathrm{hd}, \mathrm{tl})$ with $\mathrm{hd}_n : \mathrm{PStr}(A)_{(s_\infty n)} \to A$ and $\mathrm{tl}_n : \mathrm{PStr}(A)_{(s_\infty n)} \to \mathrm{PStr}(A)_n$ for every $n \in \mathbb{N}^\infty$.

We can make this construction functorial in A, using the same "trick" as for sums and products. To this end, we define the functor $H : \mathbf{P_1} \times \mathbf{P}_{\mathbb{N}^\infty} \to \mathbf{P}_{\mathbb{N}^\infty} \times \mathbf{P}_{\mathbb{N}^\infty}$ with $H = \langle !_{\mathbb{N}^\infty} \circ \pi_1, \pi_2 \rangle$, where π_1 and π_2 are corresponding projection functors, so that $H(A, X) = F(X)$. This gives, by data type completeness, rise to a functor $\nu(\widehat{G_u}, \widehat{F}) : \mathbf{P}_{\mathbb{N}^\infty} \to \mathbf{P}_{\mathbb{N}^\infty}$, which we denote by PStr, together with a pair $(\mathrm{hd}, \mathrm{tl})$ of natural transformations. □

We have seen in the examples above that we would often like to use a data type again as index, which means that we need a mechanism to turn a data type in $\mathbf{E}$ into an index in $\mathbf{B}$. This is provided by, so called, *comprehension*.

Definition 3.5 (See [17, Lem. 1.8.8, Def. 10.4.7] and [10])**.** Let $P : \mathbf{E} \to \mathbf{B}$ be a fibration. If each fibre $\mathbf{P}_I$ has a final object $\mathbf{1}_I$ and these are preserved by reindexing, then there is a fibred *final object functor* $\mathbf{1}_{(-)} : \mathbf{B} \to \mathbf{E}$. (Note that then $P(\mathbf{1}_I) = I$.) P is a *comprehension category with unit* (CCU), if $\mathbf{1}_{(-)}$ has a right adjoint $\{-\} : \mathbf{E} \to \mathbf{B}$, the *comprehension*. This gives rise to a functor $\mathscr{P} : \mathbf{E} \to \mathbf{B}^{\to}$ into the arrow category over $\mathbf{B}$, by mapping $A \mapsto P(\varepsilon_A) : \{A\} \to P(A)$, where $\varepsilon : \mathbf{1}_{\{-\}} \Rightarrow \mathrm{Id}$ is the counit of $\mathbf{1}_{(-)} \dashv \{-\}$. We often denote $\mathscr{P}(A)$ by π_A and call it the *projection* of A. Finally, P is said to be a *full* CCU, if $\mathscr{P}$ is full.

Note that, in a data type complete category, we can define final objects in each fibre, the preservation of them needs to be required separately.

Example 3.6. In Fam(**Set**), the final object functor is given by $\mathbf{1}_I = (I, \{\mathbf{1}\}_{i\in I})$, where $\mathbf{1}$ is the singleton set. Comprehension is defined to be $\{(I,X)\} = \coprod_{i\in I} X_i$ and the projections π_I map then an element of $\coprod_{i\in I} X_i$ to its component $i \in I$.

Using comprehension, we can give a general account to dependent data types.

Definition 3.7. We say that a fibration $P : \mathbf{E} \to \mathbf{B}$ is a *data type closed category* (DTCC), if it is a CCU, has a terminal object in $\mathbf{B}$ and is data type complete.

As already mentioned, the purpose of introducing comprehension is that it allows us to use data types defined in $\mathbf{E}$ again as index. The terminal object in $\mathbf{B}$ is used to introduce data types without dependencies, like the natural numbers. Let us reiterate on Ex. 3.4.

Example 3.8. Recall that we assumed the existence of extended naturals $\mathbb{N}^\infty$ and the successor map s_∞ on them to define partial streams. We are now in the position to define, in a data type closed category, everything from scratch as follows.

Having defined $+ : \mathbf{P_1} \times \mathbf{P_1} \to \mathbf{P_1}$, see Thm. 3.2, we put $\mathbb{N}^\infty = \nu(\mathrm{Id}, \mathbf{1} + \mathrm{Id})$ and find the predecessor pred as the final dialgebra on $\mathbb{N}^\infty$. The successor s_∞ arises as the coinductive extension $(\mathbb{N}^\infty, \kappa_2) \to (\mathbb{N}^\infty, \mathrm{pred})$, where κ_2 is the coproduct inclusion. Partial streams $\mathrm{PStr} : \mathbf{P}_{\{\mathbb{N}^\infty\}} \to \mathbf{P}_{\{\mathbb{N}^\infty\}}$ are then given, as in Ex. 3.4, by the final $(\widehat{G}, \widehat{F})$-dialgebra with $G = \langle \{s_\infty\}^*, \{s_\infty\}^* \rangle$ and $F = \langle !_{\mathbb{N}^\infty} \circ \pi_1, \pi_2 \rangle$. □

4 Constructing Data Types

In this section, we show how some data types can be constructed through polynomial functors, where I draw from the vast amount of work on polynomial functors that exists in the literature, see [2, 12]. The construction works by, first, reducing dialgebras to (co)algebras and, second, constructing the necessary initial algebras and final coalgebras as fixed points of polynomial functors analogously to the construction of strictly positive types in [2]. This result works thus far only for data types that, if at all, only use dependent coinductive types at the top-level. Nesting of dependent inductive and non-dependent coinductive types works, however, in full generality.

Before we come to polynomial functors and their fixed points, we show that inductive and coinductive data types actually correspond to initial algebras and final coalgebras, respectively.

Theorem 4.1. *Let $P : \mathbf{E} \to \mathbf{B}$ be a fibration with fibrewise coproducts and dependent sums. If (F,u) with $F : \mathbf{P}_I \to \mathbf{P}_{J_1} \times \cdots \times \mathbf{P}_{J_n}$ is a signature, then there is an isomorphism*

$$\mathrm{DiAlg}\,(F, G_u) \cong \mathrm{Alg}\left(\coprod_{u_1} \circ F_1 +_I \cdots +_I \coprod_{u_n} \circ F_n \right)$$

where $F_k = \pi_k \circ F$ is the kth component of F. In particular, existence of inductive data types and initial algebras coincide. Dually, if P has fibrewise and dependent products, then

$$\mathrm{DiAlg}\,(G_u, F) \cong \mathrm{CoAlg}\left(\prod_{u_1} \circ F_1 \times_I \cdots \times_I \prod_{u_n} \circ F_n\right).$$

In particular, existence of coinductive data types and final coalgebras coincide.

Proof. The first result is given by a simple application of the adjunctions $\coprod_{k=1}^{n} \dashv \Delta_n$ between the (fibrewise) coproduct and the diagonal, and $\coprod_{u_k} \dashv u_k^*$:

$$\frac{\dfrac{FX \longrightarrow G_u X \quad (\text{in } \mathbf{P}_{J_1} \times \cdots \times \mathbf{P}_{J_n})}{(\coprod_{u_1}(F_1 X), \ldots, \coprod_{u_n}(F_n X)) \longrightarrow \Delta_n X \quad (\text{in } \mathbf{P}_I^n)}}{\coprod_{k=1}^{n} \coprod_{u_k}(F_k X) \longrightarrow X \qquad (\text{in } \mathbf{P}_I)}$$

That (di)algebra homomorphisms are preserved follows at once from naturality of the used Hom-set isomorphisms. The correspondence for coinductive types follows by duality. □

To be able to reuse existing work, we work in the following with the codomain fibration $\mathrm{cod} : \mathbf{B}^{\to} \to \mathbf{B}$ for a category $\mathbf{B}$ with pullbacks. Moreover, we assume that $\mathbf{B}$ is locally Cartesian closed, which is equivalent to say that $\mathrm{cod} : \mathbf{B}^{\to} \to \mathbf{B}$ is a closed comprehension category, that is, it is a full CCU with products and coproducts, and $\mathbf{B}$ has a final object, see [17, Thm 10.5.5]. Finally, we need disjoint coproducts in $\mathbf{B}$, which gives us an equivalence $\mathbf{B}/_{I+J} \simeq \mathbf{B}/_I \times \mathbf{B}/_J$, see [17, Prop. 1.5.4].

Definition 4.2. A *dependent polynomial* P indexed by I on variables indexed by J is given by a triple of morphisms

$$J \xleftarrow{\ s\ } B \xrightarrow{\ f\ } A \xrightarrow{\ t\ } I$$

If $J = I = \mathbf{1}$, f is said to be a *(non-dependent) polynomial*. The extension of P is given by the composite

$$[\![P]\!] = \mathbf{B}/_J \xrightarrow{s^*} \mathbf{B}/_B \xrightarrow{\Pi_f} \mathbf{B}/_A \xrightarrow{\coprod_t} \mathbf{B}/_I,$$

which we denote by $[\![f]\!]$ if f is non-dependent. A functor $F : \mathbf{B}/_J \to \mathbf{B}/_I$ is a *dependent polynomial functor*, if there is a dependent polynomial P such that $F \cong [\![P]\!]$.

Remark 4.3. Note that polynomials are called *containers* by Abbott et al. [2, 1], and a polynomial $P = 1 \xleftarrow{!} B \xrightarrow{f} A \xrightarrow{!} 1$ would be written as $A \triangleright f$. Container morphisms, however, are different from those of dependent polynomials, as the latter correspond strong natural transformations [12, Prop. 2.9], whereas the former are in exact correspondence with all natural transformations between extensions [2, Thm. 3.4].

Because of this relation, we will apply results for containers that do not involve morphisms to polynomials. In particular, [2, Prop. 4.1] gives us that we can construct final coalgebras for polynomial functors from initial algebras for polynomial functors. The former are called *M-types* and are denoted by M_f for $f : A \to B$, whereas the latter are *W-types* and denoted by W_f.

Assumption 4.4. We assume that **B** is closed under the formation of W-types, thus is a *Martin-Löf category* in the terminology of [2].

By the above remark, **B** then also has all M-types.

Analogously to how [11, Thm. 12] extends [20, Prop. 3.8], we extend here [6, Thm 3.3]. As it was pointed out by one reviewer, this result is actually in [5], the published version of [6].

Theorem 4.5. *If* **B** *has finite limits, then every dependent polynomial functor has a final coalgebra in* $\mathbf{B}/_I$.

Proof. Let $P = I \xleftarrow{s} B \xrightarrow{f} A \xrightarrow{t} I$ be a dependent polynomial, we construct, analogously to [11] the final coalgebra V of $[\![P]\!]$ as an equaliser as in the following diagram, in which $f \times I$ is a shorthand for $B \times I \xrightarrow{f\times \mathrm{id}_I} A \times I$ and $M_{f\times I}$ is the carrier of the final $[\![f \times I]\!]$-coalgebra.

$$V \xrightarrow{\ g\ } M_f \underset{u_2}{\overset{u_1}{\rightrightarrows}} M_{f\times I}$$

First, we give u_1 and u_2, whose definitions are summarised in the following diagrams.

$$\begin{array}{ccc}
M_f & \xrightarrow{u_1} & M_{f\times I} \\
\downarrow{\scriptstyle \xi_f} & & \downarrow{\scriptstyle \xi_{f\times I}} \\
[\![f]\!](M_f) & & \\
\downarrow{\scriptstyle p_{M_f}} & & \\
[\![f\times I]\!](M_f) & \xrightarrow{[\![f\times I]\!](u_1)} & [\![f\times I]\!](M_{f\times I})
\end{array}
\qquad
\begin{array}{ccccc}
 & & u_2 & & \\
 & \overset{u_1}{\nearrow} & & \overset{\psi}{\searrow} & \\
M_f & \xrightarrow{u_1} & M_{f\times I} & \xrightarrow{\psi} & M_{f\times I} \\
 & & \downarrow{\scriptstyle \xi_{f\times I}} & & \downarrow{\scriptstyle \xi_{f\times I}} \\
 & & [\![f\times I]\!](M_{f\times I}) & & \\
 & & \downarrow{\scriptstyle \Sigma_{A\times I}K} & & \\
 & & [\![f\times I]\!](M_{f\times I}\times B) & \xrightarrow{[\![f\times I]\!](\phi)} & [\![f\times I]\!](M_{f\times I})
\end{array}$$

These diagrams shall indicate that u_1 is given as coinductive extensions and ψ as one-step definition (which can be defined using coproducts), using that $M_{f\times I}$ is a final coalgebra. The maps involved in the diagram are given as follows, which we sometimes spell out in the internal language of cod, see for example [1], as this is sometimes more readable.

- $p : \Sigma_A \Pi_f \Rightarrow \Sigma_{A\times I}\Pi_{f\times I}$ is the natural transformation that maps (a,v) to $(a,t(a),v)$. It is given by the extension $[\![\alpha,\beta]\!] : [\![f]\!] \Rightarrow [\![f\times I]\!]$ of the morphism of polynomials [12]

$$\begin{array}{ccc}
B & \xrightarrow{f} & A \\
{\scriptstyle\beta}\downarrow & \lrcorner & \downarrow{\scriptstyle\alpha} \\
B\times I & \xrightarrow{f\times I} & A\times I
\end{array}$$

  where $\alpha = \langle \mathrm{id}, t\rangle$ and $\beta = \langle \mathrm{id}, t\circ f\rangle$.

- The map $K : \Pi_{f\times I}(M_{f\times I}) \to \Pi_{f\times I}(M_{f\times I}\times B)$ is given as transpose of $\langle \varepsilon_{M_{f\times I}}, \pi_1 \circ \pi\rangle : (f\times I)^*(\Pi_{f\times I}(M_{f\times I})) \to M_{f\times I}\times B$, where ε is the counit of the product (evaluation) and π is the context projection. In the internal language K is given by $K\,v = \lambda(b,i).(v\,(b,i),b)$.

- $\phi : M_{f\times I} \times B \to M_{f\times I}$ is constructed as coinductive extension as in the following diagram

$$\begin{array}{ccc} M_{f\times I}\times B & \xrightarrow{\quad\phi\quad} & M_{f\times I} \\ \downarrow{\scriptstyle \xi_{f\times I}\times \mathrm{id}} & & \\ [\![f\times I]\!](M_{f\times I})\times B & & \downarrow{\scriptstyle \xi_{f\times I}} \\ \downarrow{\scriptstyle e} & & \\ [\![f\times I]\!](M_{f\times I}\times B) & \xrightarrow{[\![f\times I]\!](\phi)} & [\![f\times I]\!](M_{f\times I}) \end{array}$$

  Here e is given by $e((a,i,v),b) = (a, sb, \lambda(b', sb).(v(b',i),b'))$.

The important property, which allows us to prove that $\xi_f : M_f \to [\![f]\!](M_f)$ restricts to $\xi' : V \to [\![P]\!](V)$ and that ξ' is a final coalgebra, is that $x : V_i \iff \xi_f x = (a : A, v : \Pi_f M_f), t\,a = i$ and $(\forall b : B. f\,b = a \Rightarrow v\,b : V_{sb})$. The direction from left to right is given by simple a calculation, whereas the other direction can be proved by establishing a bisimulation and between $u_1 x$ and $u_2 x$.

Hence V, given as a subobject of M_f, is indeed the final $[\![P]\!]$-coalgebra in $\mathbf{B}/_I$. □

Combining this with [2, Prop. 4.1], we have that the existence of final coalgebras for dependent polynomial functors follows from the existence of initial algebras of (non-dependent) polynomial functors. This gives us the possibility of interpreting non-nested fixed points in any Martin-Löf category as follows.

First, we observe that the equivalence $\mathbf{B}/_{I+J} \simeq \mathbf{B}/_I \times \mathbf{B}/_J$ allows us to rewrite the functors from Thm. 4.1 to a form that is closer to polynomial functors:

$$\coprod_{u_1} \circ F_1 +_I \cdots +_I \coprod_{u_n} \circ F_n \cong \coprod_u F'$$
$$\prod_{u_1} \circ F_1 \times_I \cdots \times_I \prod_{u_n} \circ F_n \cong \prod_u F',$$

where $J = J_1 + \cdots + J_n$, $u : J \to I$ is given by the cotupling $[u_1, \ldots, u_n]$ and $F' : \mathbf{B}/_I \to \mathbf{B}/_J$ is given by $F' = \langle F_1, \ldots, F_n\rangle : \mathbf{B}/_I \to \prod_{i=1}^n \mathbf{B}/_{J_i} \simeq \mathbf{B}/_J$. Thus, if we establish that F' is a polynomial functor, we get that $\coprod_u F'$ and $\prod_u F'$ are polynomial functors, see [1]. For non-nested fixed points, that is, F_k is either a constant functor, given by composition or reindexing, this is immediate, as dependent polynomials can be composed and are closed under constant functors and reindexing, see [12].

We say that a dependent polynomial is *parametric*, if it is of the following form.

$$K + I \xleftarrow{\ s\ } B \xrightarrow{\ f\ } A \xrightarrow{\ t\ } I$$

Such polynomials represent polynomial functors $\mathbf{B}/_K \times \mathbf{B}/_I \to \mathbf{B}/_I$ and allow us speak about nested fixed points just as we have done in Sec. 2. What thus remains is that fixed points of parametric dependent polynomial functors, in the sense of Sec. 2, are again dependent polynomial functors.

The proof of this is literally the same as that for containers [1, Sec. 5.3-5.5] or non-dependent polynomials [11], except that we need to check some extra conditions regarding the indexing.

Theorem 4.6. *Initial algebras and final coalgebras of parametric, dependent polynomial functors are again dependent polynomial functors.*

Proof. Let

$$F = J \xleftarrow{s} B \xrightarrow{f} A \xrightarrow{t} I$$
$$G = I \xleftarrow{u} D \xrightarrow{g} C \xrightarrow{v} I$$

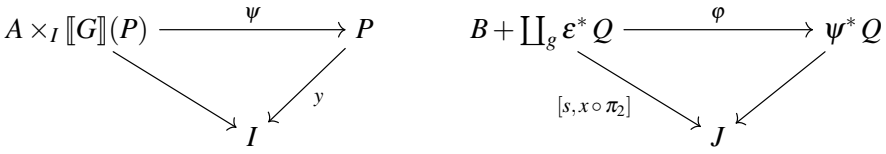
be dependent polynomials and $H(X,Y) = [\![F]\!] \times_I [\![G]\!]$ be the parametric dependent polynomial functor in question. Assuming that there is a polynomial

$$J \xleftarrow{x} Q \xrightarrow{h} P \xrightarrow{y} I$$

so that for $K = \coprod_y \prod_h x^*$ we have $K(X) \cong H(X, K(X))$, we can calculate, as in [1], that we need to have isomorphisms

$$\psi : A \times_I [\![G]\!](P) \cong P$$
$$\varphi : B + \coprod_g \varepsilon^* Q \cong \psi^*(Q)$$

where $B + \coprod_g \varepsilon^* Q$ is, as in loc. cit., is an abbreviation for $B_a + \coprod_{d:D_c} Q(rd)$ in the context $(a,(c,r)) : A \times_I [\![G]\!](P)$. If $K(X)$ shall be an initial algebra, ψ must an initial algebra as well, whereas if $K(X)$ shall be a final coalgebra, ψ must be one. The isomorphism φ is given as the initial $(\psi^{-1})^*(B + \coprod_g \varepsilon^*)$-algebra in both cases, see [1]. This we use to define $x : Q \to J$ as the inductive extension of the map $[s, \pi_2] : (\psi^{-1})^*(B + \coprod_g \varepsilon^* J) \to J$. Given these definitions, the following diagrams commute.



This gives us that the isomorphism given in the proofs of [1, Prop. 5.3.1, Prop. 5.4.2] also work for the dependent polynomial case. The rest of the proofs in loc. cit. go then through, as well. Thus K is in both cases again given by a dependent polynomial. □

Summing up, we are left with the following result.

Corollary 4.7. *All data types for strictly positive signatures can be constructed in any Martin-Löf category.*

Let us see, by means of an example, how the construction in the proof of Thm. 4.5 works intuitively.

Example 4.8. Recall from Ex. 3.4 that partial streams are given by the declaration

codata PStr (A : **Set**) : $\mathbb{N}^\infty \to$ **Set where**
  hd : (n : $\mathbb{N}^\infty$) → PStr (s_∞ n) → A
  tl : (n : $\mathbb{N}^\infty$) → PStr (s_∞ n) → PStr n

By Thm. 4.1, we can construct PStr as the final coalgebra of $F : \mathbf{B}/\mathbf{1} \times \mathbf{B}/\mathbb{N}^\infty \to \mathbf{B}/\mathbb{N}^\infty$ with $F(A,X) = \prod_{s_\infty} !^* A \times \prod_{s_\infty} X$. Note that F is isomorphic to $\mathbf{B}/\mathbf{1} \times \mathbf{B}/\mathbb{N}^\infty \simeq \mathbf{B}/\mathbf{1}+\mathbb{N}^\infty \xrightarrow{[\![P]\!]} \mathbf{B}/\mathbb{N}^\infty$, where P is the polynomial

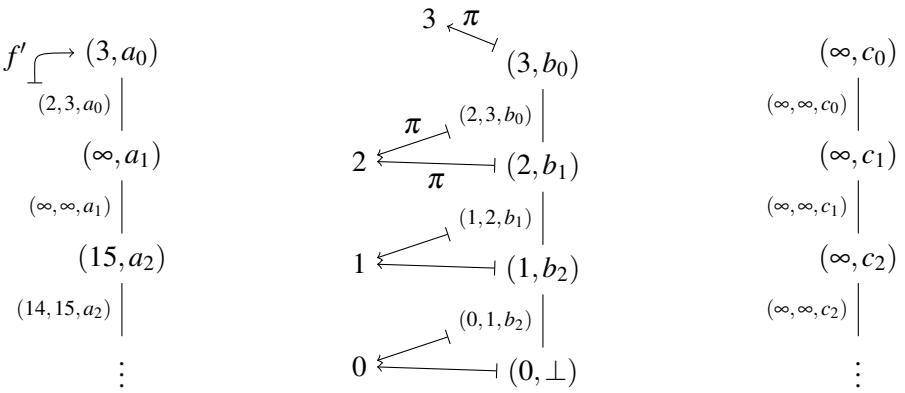
$$P = \mathbf{1} + \mathbb{N}^\infty \xleftarrow{g} 2 \times \mathbb{N}^\infty \xrightarrow{f} \mathbb{N}^\infty \xrightarrow{\mathrm{id}} \mathbb{N}^\infty \qquad g(i,k) = \begin{cases} \kappa_1 *, & i = 1 \\ \kappa_2 k, & i = 2 \end{cases} \qquad f(i,k) = s_\infty k.$$

If we now fix an object $A \in \mathbf{B}/\mathbf{1}$, then $F(A,-) \cong [\![P']\!]$ for the polynomial P' given by

$$P' = \mathbb{N}^\infty \xleftarrow{\pi} \sum_{\mathbb{N}^\infty} \sum_{s_\infty} \prod_{s_\infty} !^* A \xrightarrow{f'} \sum_{\mathbb{N}^\infty} \prod_{s_\infty} !^* A \xrightarrow{\pi} \mathbb{N}^\infty,$$

where π is the projection on the index of a dependent sum and $f'(n,(s_\infty n, v)) = (s_\infty n, v)$.

Recall that we construct in Thm. 4.5 the final coalgebra of $[\![P']\!]$ as a subobject of $M_{f'}$. Below, we present three trees that are elements of $M_{f'}$, where only the second and third are actually selected by the equaliser taken in Thm. 4.5.



Here we denote a pair $(k,v) : \sum_{\mathbb{N}^\infty} \prod_{s_\infty} !^* A$ with $k = s_\infty n$ and $vn = a$ by (k,a), or if $k = 0$ by $(0,\bot)$. Moreover, we indicate the matching of indices in the second tree, which is used to form the equaliser. Note that the second tree is an element of $\mathrm{PStr}(A)\,3$, whereas the third is in $\mathrm{PStr}(A)\,\infty$. □

5 Conclusion and Future Work

We have seen how dependent inductive and coinductive types with type constructors, in the style of Agda, can be given semantics in terms of data type closed categories (DTCC), with the restriction that destructors of coinductive types are not allowed to refer to each other. This situation is summed up in the following table.

Condition	Use/Implications
Cloven fibration	Definition of signatures and data types
Data type completeness	Construction of types indexed by objects in base (e.g., vectors for $\mathbb{N} \in \mathbf{B}$) and types agnostic of indices (e.g., initial and final objects, sums and products)
Data type closedness	Constructed types as index; Full interpretation of data types

Moreover, we have shown that a large part of these data types can be constructed as fixed points of polynomial functors.

Let us finish by discussing directions for future work. First, a full interpretation of syntactic data types has also still to be carried out. Here one has to be careful with type equality, which is usually dealt with using split fibrations and a Beck-Chevalley condition. The latter can be defined generally for the data types of this work, in needs to be checked, however, whether this condition is sufficient for giving a sound interpretation. Finally, the idea of using dialgebras has found its way into the syntax of higher inductive types [7], though in that work the used format of dialgebras is likely to be too liberal to

guarantee the existence of semantics. The reason is that the shape of dialgebras used in the present work ensures that we can construct data types from (co)coalgebras, whereas this is not the case in [7]. Thus it is to be investigated what the right notion of dialgebras is for capturing higher (co)inductive types, such that their semantics in terms of trees can always be constructed.

References

[1] Michael Abbott (2003): *Categories of Containers*. Ph.D. thesis, Leicester.

[2] Michael Abbott, Thorsten Altenkirch & Neil Ghani (2005): *Containers: Constructing strictly positive types*. *Theoretical Computer Science* 342(1), pp. 3–27, doi:10.1016/j.tcs.2005.06.002.

[3] Thorsten Altenkirch & Peter Morris (2009): *Indexed containers*. In: *Logic In Computer Science, 2009. LICS'09. 24th Annual IEEE Symposium on*, IEEE, pp. 277–285, doi:10.1109/LICS.2009.33.

[4] Thorsten Altenkirch, Peter Morris, Fredrik Nordvall Forsberg & Anton Setzer (2011): *A Categorical Semantics for Inductive-Inductive Definitions*. In Andrea Corradini, Bartek Klin & Corina Cîrstea, editors: *Algebra and Coalgebra in Computer Science*, *Lecture Notes in Computer Science* 6859, Springer Berlin Heidelberg, pp. 70–84, doi:10.1007/978-3-642-22944-2_6.

[5] Benno van den Berg & Federico De Marchi (2007): *Non-well-founded trees in categories*. *Annals of Pure and Applied Logic* 146(1), pp. 40–59, doi:10.1016/j.apal.2006.12.001.

[6] Benno van den Berg & Federico de Marchi (2004): *Non-well-founded trees in categories*. *arXiv:math/0409158*. Available at `http://arxiv.org/abs/math/0409158`.

[7] Paolo Capriotti (2014): *Mutual and Higher Inductive Types in Homotopy Type Theory*. Available at `http://www.cs.nott.ac.uk/~pvc/away-day-2014/mhit.pdf`.

[8] James Chapman, Pierre-Évariste Dagand, Conor McBride & Peter Morris (2010): *The Gentle Art of Levitation*. In: *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP '10, ACM, New York, NY, USA, pp. 3–14, doi:10.1145/1863543.1863547.

[9] P.-E. Dagand & C. McBride (2013): *A Categorical Treatment of Ornaments*. In: *2013 28th Annual IEEE/ACM Symposium on Logic in Computer Science (LICS)*, pp. 530–539, doi:10.1109/LICS.2013.60.

[10] Clément Fumex, Neil Ghani & Patricia Johann (2011): *Indexed Induction and Coinduction, Fibrationally*. In Andrea Corradini, Bartek Klin & Corina Cîrstea, editors: *Algebra and Coalgebra in Computer Science*, *LNCS* 6859, Springer, pp. 176–191, doi:10.1007/978-3-642-22944-2_13.

[11] Nicola Gambino & Martin Hyland (2004): *Wellfounded Trees and Dependent Polynomial Functors*. In: *Types for Proofs and Programs*, *LNCS* 3085, Springer, pp. 210–225, doi:10.1007/978-3-540-24849-1_14.

[12] Nicola Gambino & Joachim Kock (2013): *Polynomial functors and polynomial monads*. *Math. Proc. Camb. Philos. Soc.* 154(01), pp. 153–192, doi:10.1017/S0305004112000394. Available at `http://arxiv.org/abs/0906.4931`.

[13] Tatsuya Hagino (1987): *A typed lambda calculus with categorical type constructors*. In: *Category Theory in Computer Science*, pp. 140–157.

[14] Makoto Hamana & Marcelo Fiore (2011): *A Foundation for GADTs and Inductive Families: Dependent Polynomial Functor Approach*. In: *Proceedings of the Seventh Workshop on Generic Programming*, WGP '11, ACM, New York, NY, USA, pp. 59–70, doi:10.1145/2036918.2036927.

[15] Claudio Hermida & Bart Jacobs (1997): *Structural Induction and Coinduction in a Fibrational Setting*. *Inf. Comput.* 145, pp. 107–152, doi:10.1006/inco.1998.2725.

[16] Martin Hofmann (1995): *On the Interpretation of Type Theory in Locally Cartesian Closed Categories*. In: *Proceedings of Computer Science Logic, 8th Workshop, CSL'94, Selected Papers*, *LNCS* 933, Springer, pp. 427–441, doi:10.1007/BFb0022273.

[17] B. Jacobs (1999): *Categorical Logic and Type Theory*. Studies in Logic and the Foundations of Mathematics 141, North Holland, Amsterdam.

[18] Jiho Kim (2010): *Higher-order Algebras and Coalgebras from Parameterized Endofunctors*. Electronic Notes in Theoretical Computer Science 264(2), pp. 141–154, doi:10.1016/j.entcs.2010.07.018.

[19] Andres Löh & José Pedro Magalhães (2011): *Generic programming with indexed functors*. In: Proceedings of the seventh ACM SIGPLAN workshop on Generic programming, ACM, pp. 1–12. Available at `http://dl.acm.org/citation.cfm?id=2036920`.

[20] Ieke Moerdijk & Erik Palmgren (2000): *Wellfounded trees in categories*. Annals of Pure and Applied Logic 104(1–3), pp. 189–218, doi:10.1016/S0168-0072(00)00012-9.

[21] R. A. G. Seely (1984): *Locally cartesian closed categories and type theory*. Math. Proc. Camb. Philos. Soc. 95(01), pp. 33–48, doi:10.1017/S0305004100061284.