Wellfounded Trees and Dependent Polynomial Functors

Nicola Gambino* and Martin Hyland

Department of Pure Mathematics and Mathematical Statistics
University of Cambridge
{N.Gambino,M.Hyland}@dpmms.cam.ac.uk

Abstract. We set out to study the consequences of the assumption of types of wellfounded trees in dependent type theories. We do so by investigating the categorical notion of wellfounded tree introduced in [16]. Our main result shows that wellfounded trees allow us to define initial algebras for a wide class of endofunctors on locally cartesian closed categories.

1 Introduction

Types of wellfounded trees, or $\mathcal{W}$-types, are one of the most important components of Martin-Löf's dependent type theories. First, they allow us to define a wide class of inductive types [5,15]. Secondly, they play an essential role in the interpretation of constructive set theories in dependent type theories [3]. Finally, from the proof-theoretic point of view, they represent the paradigmatic example of a generalised inductive definition and contribute considerably to the proof-theoretic strength of dependent type theories [8].

In [16] a categorical counterpart of the notion of $\mathcal{W}$-type was introduced. In a locally cartesian closed category, $\mathcal{W}$-types are defined as the initial algebras for endofunctors of a special kind, to which we shall refer here as *polynomial functors*. The purpose of this paper is to study polynomial endofunctors and $\mathcal{W}$-types more closely. In particular, we set out to explore some of the consequences of the assumption that a locally cartesian closed category has $\mathcal{W}$-types, i.e. that every polynomial endofunctor has an initial algebra. To explore these consequences we introduce *dependent polynomial functors*, that generalize polynomial functors.

Our main theorem then shows that the assumption of $\mathcal{W}$-types is sufficient to define explicitly initial algebras for dependent polynomial functors. We expect this result to lead to further insight into the interplay between dependent type theory and the theory of inductive definitions. In this paper, we will limit ourselves to giving only two applications of our main theorem. First, we show how the class of polynomial functors is closed under fixpoints. We hasten to point out that related results appeared in [1,2]. One of our original goals was indeed to put those results in a more general context and simplify their proofs.

* EPSRC Postdoctoral Research Fellow in Mathematics.

S. Berardi, M. Coppo, and F. Damiani (Eds.): TYPES 2003, LNCS 3085, pp. 210–225, 2004.

Secondly, we show how polynomial functors have free monads, and these free monads are themselves polynomial. The combination of these two facts leads to further observations concerning the categories of algebras of polynomial endofunctors. These results are relevant for our ongoing research on 2-categorical models of the differential λ-calculus [6].

The interplay between dependent type theories and categories is here exploited twice. On the one hand, category theory provides a mathematically efficient setting to present results that apply not only to the categories arising from the syntax of dependent type theories, but also to the categories providing their models. On the other hand, dependent type theories provide a convenient language to manipulate and describe the objects and the arrows of locally cartesian closed categories via the internal language of such a category [18].

In order to set up the internal language for a locally cartesian closed category with $\mathcal{W}$-types, it is necessary to establish some technical results that ensure a correct interaction between the structural rules of the internal language and the rules for $\mathcal{W}$-types. Although these results are already contained in [16] we give new and simpler proofs of some of them. Once this is achieved, we can freely exploit the internal language to prove the consequences of the assumption of $\mathcal{W}$-types in a category.

2 Polynomial Functors

2.1 Locally Cartesian Closed Categories

We say that a category $\mathcal{C}$ is a *locally cartesian closed category*, or a lccc for short, if for every object I of $\mathcal{C}$ the slice category $\mathcal{C}/I$ is cartesian closed[1]. Note that if $\mathcal{C}$ is a lccc then so are all its slices. For an arrow $f : B \to A$ in a lccc $\mathcal{C}$ we write $\Delta_f : \mathcal{C}/A \to \mathcal{C}/B$ for the associated pullback functor, which can be defined since slice categories have cartesian products. The key fact about locally cartesian closed categories is the following proposition [7].

Proposition 1. *Let $\mathcal{C}$ be a lccc. For any arrow $f : B \to A$ in $\mathcal{C}$, the pullback functor $\Delta_f : \mathcal{C}/A \to \mathcal{C}/B$ has both a left and a right adjoint.*

Given an arrow $f : B \to A$ in a lccc $\mathcal{C}$, we will write $\Sigma_f : \mathcal{C}/B \to \mathcal{C}/A$ and $\Pi_f : \mathcal{C}/B \to \mathcal{C}/A$ for the left and right adjoint to the pullback functor, respectively. We indicate the existing adjunctions as $\Sigma_f \dashv \Delta_f \dashv \Pi_f$.

An abuse of language. For an arrow $f : B \to A$, we write the image of $X \to A$ in $\mathcal{C}/A$ under Δ_f as $\Delta_f(X) \to B$. These arrows fit into the pullback diagram

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

[1] Here and in the following, when we require the existence of some structure in a category, we always mean that this structure is given to us by an explicitly defined operation.

The Beck-Chevalley condition. The Beck-Chevalley condition, which holds in any lccc, expresses categorically that substitution behaves correctly with respect to type-formation rules. More precisely, it asserts that for a pullback diagram of the form

$$\begin{tikzcd} D \arrow[r, "k"] \arrow[d, "g"] & B \arrow[d, "f"] \\ C \arrow[r, "h"] & A \end{tikzcd}$$

the canonical natural transformations $\Sigma_g \; \Delta_k \Rightarrow \Delta_h \; \Sigma_f$ and $\Delta_h \; \Pi_f \Rightarrow \Pi_g \; \Delta_k$ are isomorphisms.

The axiom of choice. The type-theoretic axiom of choice [15] is expressed by the fact that, for two arrows $g : C \to B$ and $f : B \to A$, the canonical natural transformation $\Sigma_h \; \Pi_p \; \Delta_{\varepsilon_C} \Rightarrow \Pi_f \; \Sigma_g$ is an isomorphism, where

$$\begin{tikzcd} \Delta_f \Pi_f C \arrow[r, "p"] \arrow[d, "q"'] & \Pi_f C \arrow[d, "h"] \\ B \arrow[r, "f"] & A \end{tikzcd}$$

is a pullback diagram and $\varepsilon_C : \Delta_f \; \Pi_f \, C \to C$ is a component of the counit of the adjunction $\Delta_f \dashv \Pi_f$.

2.2 Internal Language

Associated to a lccc $\mathcal{C}$ there is a dependent type theory $\mathsf{Th}(\mathcal{C})$ to which we shall refer as the *internal language* of $\mathcal{C}$. A complete presentation of such a dependent type theory can be found in [9,14,18]. We limit ourselves to recalling only those aspects that are most relevant for the remainder of this work. The standard judgement forms, written here as

$$(B_a \mid a \in A)\, , \; (B_a = B'_a \mid a \in A)\, , \; (b_a \in B_a \mid a \in A)\, , \; (b_a = b'_a \in B_a \mid a \in A),$$

are assumed to have their usual meaning [15]. The dependent type theory $\mathsf{Th}(\mathcal{C})$ has the following primitive forms of type:

$$1\, , \quad Id_A(a, a')\, , \quad \sum_{a \in A} B_a\, , \quad \prod_{a \in A} B_a\; .$$

We refer to these as the unit, identity, dependent sum and dependent product types, respectively. As usual, these primitive forms of type allow us to define the forms of type $A \times B$ and B^A , to which we refer as the product and function types. The dependent type theory $\mathsf{Th}(\mathcal{C})$ has a straightforward interpretation in $\mathcal{C}$ and thus provides a convenient language to define objects and arrows in $\mathcal{C}$.

2.3 Polynomial Functors

Let $\mathcal{C}$ be a lccc. For an object I of $\mathcal{C}$, we write I also for the unique arrow $I \to 1$ into the terminal object 1 of $\mathcal{C}$. Observe this arrow determines functors $\Delta_I : \mathcal{C} \to \mathcal{C}/I$ and $\Sigma_I : \mathcal{C}/I \to \mathcal{C}$. We are now ready to introduce polynomial functors. For an arrow $f : B \to A$ in $\mathcal{C}$, we define a functor $\mathcal{P}_f : \mathcal{C} \to \mathcal{C}$, called the *generalized polynomial functor associated to* f, as the composite

$$\begin{tikzcd} \mathcal{C} \arrow[r, "\Delta_B"] & \mathcal{C}/B \arrow[r, "\Pi_f"] & \mathcal{C}/A \arrow[r, "\Sigma_A"] & \mathcal{C} \end{tikzcd} \qquad (1)$$

Definition 2. We say that $P : \mathcal{C} \to \mathcal{C}$ is a *generalized polynomial functor* if it is naturally isomorphic to a functor $\mathcal{P}_f : \mathcal{C} \to \mathcal{C}$ defined as $\mathcal{P}_f =_{\mathrm{df}} \Sigma_A \, \Pi_f \, \Delta_B$, for some arrow $f : B \to A$ of $\mathcal{C}$.

Note. To avoid clashes with the existing terminology, we adopted the name generalised polynomial functors. This is in analogy with the distinction between generalised inductive definitions and ordinary ones. Since in this paper we only consider polynomial functors in the generalised sense, we refer to them simply as polynomial functors.

Let us look more closely at the definition of polynomial endofunctors. For an arrow $f : B \to A$, we have the two functors $\Sigma_A : \mathcal{C}/A \to \mathcal{C}$ and $\Delta_B : \mathcal{C} \to \mathcal{C}/B$. The functor Δ_B takes an object X of $\mathcal{C}$ to the left-hand side of the pullback diagram:

$$\begin{tikzcd} X \times B \arrow[r] \arrow[d] & X \arrow[d, "X"] \\ B \arrow[r, "B"] & 1 \end{tikzcd}$$

We can therefore write $\Delta_B X = X \times B$. The action of Σ_A is very simple: given an object $Y \to A$ of $\mathcal{C}/A$ we have $\Sigma_A(Y \to A) = Y$. These observations lead to a description of polynomial functors in the internal language, which we shall exploit. The object $f : B \to A$ of $\mathcal{C}/A$ determines the judgement $(B_a \mid a \in A)$ of $\mathsf{Th}(\mathcal{C})$. We can then explicitly define in $\mathsf{Th}(\mathcal{C})$

$$\mathcal{P}_f(X) =_{\mathrm{df}} \sum_{a \in A} X^{B_a} ,$$

for a type X. The interpretation in $\mathcal{C}$ of the right-hand side of the definition is indeed $\mathcal{P}_f(X)$, as defined in (1).

2.4 Basic Properties of Polynomial Functors

Proposition 3. *The composition of two polynomial functors is polynomial.*

Proof. A proof using the internal language is in [2], but it is also possible to give a diagrammatic one. In either case one uses crucially the axiom of choice. □

We now assume that the lccc $\mathcal{C}$ has finite disjoint coproducts. As pullback functors are left adjoints, these finite coproducts are preserved under pullbacks and $\mathcal{C}$ has stable disjoint coproducts. They can be represented in a familiar way in the internal language $\mathsf{Th}(\mathcal{C})$, which now has also the primitive forms of type 0 and $A + B$ called the empty and disjoint sum types, respectively.

The class of polynomial functors is closed under a further operation, that will be very important in the following. To discuss it, let us introduce a family of functors $P_X : \mathcal{C} \to \mathcal{C}$, for X in $\mathcal{C}$, associated to a functor $P : \mathcal{C} \to \mathcal{C}$. First of all, observe that the function mapping (X, Y) into $X + PY$ can be extended to a bifunctor $\mathcal{C} \times \mathcal{C} \to \mathcal{C}$. This determines a functor $\mathcal{C} \to \mathsf{End}(\mathcal{C})$ mapping X into P_X, whose action on Y in $\mathcal{C}$ is defined by letting $P_X(Y) = X + PY$.

Proposition 4. *Let $P : \mathcal{C} \to \mathcal{C}$ be a functor and X be an object of $\mathcal{C}$. If P is polynomial then so is P_X.*

Proof. We give a proof using the internal language. Let $f : B \to A$ and consider the polynomial functor $\mathcal{P}_f$ associated to f. For X and Y in $\mathcal{C}$ we then have

$$X + \mathcal{P}_f(Y) = X + \sum_{a \in A} Y^{B_a} \cong \sum_{z \in X + A} Y^{B_z}$$

where $(B_z \mid z \in X + A)$ is defined so that the judgements $(B_{\iota_1(x)} = 0 \mid x \in X)$ and $(B_{\iota_2(a)} = B_a \mid a \in A)$ are derivable. □

To recall the notions of strength for a functor, let us consider a monoidal category $(\mathcal{C}, \otimes, I, a, l, r)$, where I is the unit object and a, l, r are natural isomorphisms giving the associativity, left and right unit laws and satisfying the monoidal coherence axioms [12]. We can regard a lccc $\mathcal{C}$ as a monoidal category where cartesian product is the tensor, and the terminal object is the unit.

Definition 5. Let $P : \mathcal{C} \to \mathcal{C}$ be a functor. By a *strength* for P we mean a natural transformation σ with components $\sigma_{X,Y} : X \otimes PY \to P(X \otimes Y)$, for X and Y in $\mathcal{C}$, such that for all X, Y, Z in $\mathcal{C}$ the following equations hold:

$$P(l_X) \circ \sigma_{X,I} = l_X \,, \quad P(r_Y) \circ \sigma_{I,Y} = r_Y \,, \quad \sigma_{X,Y \otimes Z} \circ (1_X \otimes \sigma_{Y,Z}) = \sigma_{X \times Y, Z} \,.$$

Proposition 6. *Every polynomial functor has a strength.*

Proof. Let us use the internal language to define the arrow

$$\sigma_{X,Y} : X \times \mathcal{P}_f Y \to \mathcal{P}_f(X \times Y)$$

which gives us one of the components of the required strength σ for a polynomial functor $\mathcal{P}_f$. First, observe that the domain and the codomain of $\sigma_{X,Y}$ can be described in $\mathsf{Th}(\mathcal{C})$ as $X \times \sum_{a \in A} Y^{B_a}$ and $\sum_{a \in A} (X \times Y)^{B_a}$ respectively. We can then define $\sigma_{X,Y}$ by letting $\sigma_{X,Y}(x, a, t) =_{\mathrm{df}} \big(a, (\lambda b \in B_a)(x, t(b))\big)$ for $(x, a, t) \in X \times \sum_{a \in A} Y^{B_a}$. □

3 Change of Base

In the following, we shall be interested in the effect that pullback functors have on algebras for polynomial endofunctors. Let us first recall some basic definitions. Let $P : \mathcal{C} \to \mathcal{C}$ be an endofunctor on a category $\mathcal{C}$. An *algebra* for P, or a P-algebra, is a diagram of the form $x : PX \to X$ in $\mathcal{C}$. An arrow of P-algebras from $PX \to X$ to $PY \to Y$ is given by a commuting diagram of form

$$\begin{tikzcd}
PX \arrow[r, "Pf"] \arrow[d] & PY \arrow[d] \\
X \arrow[r, "f"] & Y
\end{tikzcd}$$

There is then a manifest category P-*alg* of P-algebras and P-algebra arrows. We write $U : P\text{-}alg \to \mathcal{C}$. for the obvious forgetful functor.

In the following, we will work in a lccc $\mathcal{C}$. For an arrow $u : I \to J$ in $\mathcal{C}$ we will show that the algebras for the polynomial functor $\mathcal{P}_f$ on $\mathcal{C}/J$ associated to an arrow f of $\mathcal{C}/J$ can be mapped functorially into algebras for the polynomial endofunctor $\mathcal{P}_{\Delta_u(f)}$ on $\mathcal{C}/I$ associated to the arrow $\Delta_u(f)$ of $\mathcal{C}/I$. As we will see, this is a purely formal consequence of some observations concerning the 2-category of polynomial functors, that we define below. The treatment is inspired by the formal theory of monads [10,19].

3.1 The 2-Category of Polynomial Functors

Let us define the 2-category Poly. An object of Poly is a pair $(\mathcal{C}, \mathcal{P}_f)$ where $\mathcal{C}$ is a lccc and $\mathcal{P}_f$ is the polynomial endofunctor on $\mathcal{C}$ associated to an arrow $f : B \to A$ in $\mathcal{C}$. A 1-cell with domain $(\mathcal{C}, \mathcal{P}_f)$ and codomain $(\mathcal{D}, \mathcal{P}_g)$ is given by a pair (F, ϕ) where $F : \mathcal{C} \to \mathcal{D}$ is a functor and $\phi : \mathcal{P}_g\ F \Rightarrow F\ \mathcal{P}_f$, is a natural transformation, usually drawn in a diagram of form

$$\begin{tikzcd}
 & \mathcal{C} \arrow[dl, "F"'] \arrow[dr, "\mathcal{P}_f"] & \\
\mathcal{D} \arrow[dr, "\mathcal{P}_g"'] & \Longrightarrow & \mathcal{C} \arrow[dl, "F"] \\
 & \mathcal{D} &
\end{tikzcd}$$

The 2-cells of Poly are defined exactly as the ones in 2-categories of monads [19]. We can now define a 2-functor $\mathsf{Alg} : \mathsf{Poly} \to \mathsf{Cat}$, but for the purposes of this paper, it is sufficient to give the definition of its action on objects and 1-cells. For an object $(\mathcal{C}, \mathcal{P}_f)$ of Poly we define

$$\mathsf{Alg}(\mathcal{C}, \mathcal{P}_f) =_{\mathrm{df}} \mathcal{P}_f\text{-}alg\,.$$

Given a 1-cell $(F, \phi) : (\mathcal{C}, \mathcal{P}_f) \to (\mathcal{D}, \mathcal{P}_g)$ in Poly, the functor $\mathsf{Alg}(F, \phi)$ is defined by mapping a $\mathcal{P}_f$-algebra $x : \mathcal{P}_f\, X \to X$ into the the composite

$$\mathcal{P}_g\, F\, X \xrightarrow{\phi_X} F\, \mathcal{P}_f\, X \xrightarrow{Fx} F\, X$$

that is a $\mathcal{P}_g$-algebra.

3.2 Pullback of Algebras

By a *locally cartesian closed functor*, or lccc functor, we mean a functor that preserves the locally cartesian closed structure up to isomorphism. The next proposition is a simple but useful fact.

Proposition 7. *Let $\mathcal{C}$ and $\mathcal{D}$ be lccc's, and let $F : \mathcal{C} \to \mathcal{D}$ be a lccc functor. For any arrow $f : B \to A$ there is a natural isomorphism*

$$\chi_F : \mathcal{P}_{Ff}\, F \Rightarrow F\, \mathcal{P}_f$$

such that the 1-cell $(F, \chi_F) : (\mathcal{C}, \mathcal{P}_f) \to (\mathcal{D}, \mathcal{P}_{Ff})$ determines a commuting diagram of form

$$\begin{tikzcd}
\mathcal{P}_f\text{-}alg \arrow[r] \arrow[d, "\mathsf{Alg}(F, \chi_F)"'] & \mathcal{C} \arrow[d, "F"] \\
\mathcal{P}_{Ff}\text{-}alg \arrow[r] & \mathcal{D}
\end{tikzcd}$$

where the horizontal arrows are the forgetful functors.

Proof. For an arrow $f : B \to A$ the required natural isomorphism χ_F is obtained by pasting the three isomorphims in the diagram

$$\begin{tikzcd}
\mathcal{C} \arrow[r, "\Delta_B"] \arrow[d, "F"'] \arrow[dr, phantom, "\cong"] & \mathcal{C}/B \arrow[r, "\Pi_f"] \arrow[d, "F/B"] \arrow[dr, phantom, "\cong"] & \mathcal{C}/A \arrow[r, "\Sigma_A"] \arrow[d, "F/A"] \arrow[dr, phantom, "\cong"] & \mathcal{C} \arrow[d, "F"] \\
\mathcal{D} \arrow[r, "\Delta_{FB}"'] & \mathcal{D}/FB \arrow[r, "\Pi_{Ff}"'] & \mathcal{D}/FA \arrow[r, "\Sigma_{FA}"'] & \mathcal{D}
\end{tikzcd}$$

where for an object I, we write $F/I : \mathcal{C}/I \to \mathcal{D}/FI$ for the obvious functor induced by F. The isomorphisms in the diagram exist since F is a lccc functor. The rest of the proof follows by direct calculation. □

We can apply Proposition 7 to pullback functors, as they are lccc functors [7].

Corollary 8. *Let $\mathcal{C}$ be a lccc. Let I be an object of $\mathcal{C}$. For an arrow $f : B \to A$ in $\mathcal{C}$ there is a natural isomorphism $\chi_{\Delta_I} : \mathcal{P}_g\, \Delta_I \Rightarrow \Delta_I\, \mathcal{P}_f$, where $g =_{\mathrm{df}} \Delta_I f$.*

4 Wellfounded Trees

Definition 9. We say that a lccc $\mathcal{C}$ *has* $\mathcal{W}$*-types* if for every arrow $f : B \to A$ in $\mathcal{C}$ there is a diagram $\mathcal{P}_f(\mathcal{W}_f) \to \mathcal{W}_f$ which is an initial algebra for $\mathcal{P}_f : \mathcal{C} \to \mathcal{C}$.

Recall that, by a theorem of Lambek, the arrow $\mathcal{P}_f(\mathcal{W}_f) \to \mathcal{W}_f$ is an isomorphism. Once the internal language of a lccc with $\mathcal{W}$-types is set up, we will therefore be allowed to write

$$W \cong \sum_{a \in A} W^{B_a}$$

where $f : B \to A$ and $W =_{\mathrm{df}} \mathcal{W}_f$. The next subsection is devoted to justify the use of the internal language in connection to $\mathcal{W}$-types.

4.1 Pullback of Wellfounded Trees

In [16] it is proved that if $\mathcal{C}$ has $\mathcal{W}$-types then so do all its slices. A proof of this fact can be obtained by defining explicitly initial algebras for polynomial endofunctors on the slice categories. It is also observed there that the pullback functors preserve $\mathcal{W}$-types. Although in [16] it is suggested to prove this second fact using the explicit definition of $\mathcal{W}$-types in slice categories, we give a new and more direct proof of this fact.

Let $\mathcal{C}$ be a lccc and let I be an object in $\mathcal{C}$. Recall from Corollary 8 that there is a natural isomorphism $\chi_{\Delta_I} : \mathcal{P}_g\, \Delta_I \Rightarrow \Delta_I\, \mathcal{P}_f$ where $g =_{\mathrm{df}} \Delta_I(f)$. This natural transformation determines a functor $F_I : \mathcal{P}_f\text{-}alg \to \mathcal{P}_g\text{-}alg$ defined as $F_I =_{\mathrm{df}} \mathsf{Alg}(\Delta_I, \chi_{\Delta_I})$. We now use the inverse to χ_{Δ_I}, given by a natural transformation $\psi : \Delta_I\, \mathcal{P}_f \Rightarrow \mathcal{P}_g\, \Delta_I$, to define a functor $G_I : \mathcal{P}_g\text{-}alg \to \mathcal{P}_f\text{-}alg$ that is right adjoint to F_I. First of all, observe that ψ gives us a natural transformation $\xi : \mathcal{P}_f\, \Pi_I \Rightarrow \Pi_I\, \mathcal{P}_g$ that is defined as the composite

$$\mathcal{P}_f \Pi_I \xRightarrow{\eta\, \mathcal{P}_f\, \Pi_I} \Pi_I\, \Delta_I \mathcal{P}_f\, \Pi_I \xRightarrow{\Pi_I\, \psi\, \Pi_I} \Pi_I\, \mathcal{P}_g \Delta_I\, \Pi_I \xRightarrow{\Pi_I\, \mathcal{P}_g\, \varepsilon} \Pi_I \mathcal{P}_g$$

where η and ε are the unit and the counit of the adjunction $\Delta_I \dashv \Pi_I$, respectively. Hence we have that $(\Pi_I, \xi) : (\mathcal{C}/I, \mathcal{P}_g) \to (\mathcal{C}, \mathcal{P}_f)$ is a 1-cell in Poly and thus we can simply define $G_I =_{\mathrm{df}} \mathsf{Alg}(\Pi_I, \xi)$.

Theorem 10. *Let $\mathcal{C}$ be a lccc and let $f : B \to A$ be an arrow in $\mathcal{C}$. For any object I of $\mathcal{C}$ the adjunction $\Delta_I \dashv \Pi_I$ lifts to an adjunction $F_I \dashv G_I$, i.e. in the diagram*

$$\begin{tikzcd}
\mathcal{P}_g\text{-}alg \arrow[r] \arrow[d, shift left, "G_I"] & \mathcal{C}/I \arrow[d, shift left, "\Pi_I"] \\
\mathcal{P}_f\text{-}alg \arrow[r] \arrow[u, shift left, "F_I"] & \mathcal{C} \arrow[u, shift left, "\Delta_I"]
\end{tikzcd}$$

where $g =_{\mathrm{df}} \Delta_I(f)$, the inner and outer squares commute.

The functor G_I can be described also in the internal language. Let us consider a $\mathcal{P}_g$-algebra, i.e. an arrow $z : \mathcal{P}_g Z \to Z$ in $\mathcal{C}/I$. This arrow determines the judgement

$$\big(\ z(i,(a,s)) \in Z_i \mid i \in I\,, (a,s) \in \sum_{a \in A} {Z_i}^{B_a}\ \big)$$

where $(Z_i \mid i \in I)$ is the judgement associated to the object $Z \to I$ of $\mathcal{C}/I$. We can then derive the judgement

$$\big(\ (\lambda i \in I)\, z(i,(a,(\lambda b \in B_a)\, t(b,i))) \in \prod_{i \in I} Z_i \mid (a,t) \in \sum_{a \in A} \prod_{i \in I} {Z_i}^{B_a}\big)$$

which gives us a $\mathcal{P}_f$-algebra $\mathcal{P}_f \Pi_I Z \to \Pi_I Z$. This is exactly the image under G_I of the $\mathcal{P}_f$-algebra $\mathcal{P}_g Z \to Z$. A proof of Theorem 10 can then be obtained either reasoning with diagrams or with the internal language. We can now derive a simple proof of the pullback stability for $\mathcal{W}$-types.

Corollary 11 (Pullbacks preserve $\mathcal{W}$-types). *Let $\mathcal{C}$ be a lccc. Let $u : I \to J$ be an arrow in $\mathcal{C}$. For objects $B \to J$ and $A \to J$ in $\mathcal{C}/J$ and an arrow $f : B \to A$ between them, there is an isomorphism $\mathcal{W}_{\Delta_u(f)} \cong \Delta_u \mathcal{W}_f$.*

Proof. Note that without loss of generality we can assume that J is the terminal object of $\mathcal{C}$ and thus consider the pullback functors $\Delta_I : \mathcal{C} \to \mathcal{C}/I$ determined by $I : I \to 1$. But now it suffices to appy Theorem 10 and observe that F_I, defined as $F_I =_{\mathrm{df}} \mathsf{Alg}(\Delta_I, \chi_{\Delta_I})$, preserves initial objects because it is a left adjoint. □

5 Dependent Polynomial Functors

We can now pick up the fruits of the work done in the last section and exploit freely the internal language to prove further consequences of the assumption of the existence of $\mathcal{W}$-types in a lccc. Here we show how $\mathcal{W}$-types can be used to define initial algebras for a class of functors that is wider than the one of polynomial functors. Let $\mathcal{C}$ be a lccc. For a diagram, which we do *not* assume to be commuting, of form

$$\begin{tikzcd} B \arrow[rr, "f"] \arrow[dr, "s"'] & & A \arrow[dl, "r"] \\ & I & \end{tikzcd} \tag{2}$$

we define $\mathcal{D} : \mathcal{C}/I \to \mathcal{C}/I$, called the *dependent polynomial endofunctor* associated to the diagram, as the composite

$$\mathcal{C}/I \xrightarrow{\Delta_s} \mathcal{C}/B \xrightarrow{\Pi_f} \mathcal{C}/A \xrightarrow{\Sigma_r} \mathcal{C}/I$$

We can describe the action of $\mathcal{D}$ on an object $(X_i \mid i \in I)$ of $\mathcal{C}/I$ by letting

$$\mathcal{D}(X_i \mid i \in I) =_{\mathrm{df}} \big(\sum_{a \in A_i} \prod_{b \in B_a} X_{sb} \mid i \in I \big) \tag{3}$$

for an object $(X_i \mid i \in I)$ of $\mathcal{C}/I$. Using this description, we can observe that initial algebras for dependent polynomial functors are categorical counterparts of the so-called *general trees* of Martin-Löf type theory, as described in [17, Chapter 16]. We now give some examples of dependent polynomial functors.

(i) Polynomial functors on slice categories are special examples of dependent polynomial functors. Observe that, if the diagram in (2) commmutes, then the formula in (3) simplifies to

$$\mathcal{D}(X_i \mid i \in I) = \big(\sum_{a \in A_i} X_i^{B_a} \mid i \in I \big) .$$

(ii) Let $f : B \to A$ be an arrow in $\mathcal{C}$ and define $W =_{\mathrm{df}} \mathcal{W}_f$. For our applications, it is useful to observe that, for an arrow $g : C \to B$, the endofunctor $F : \mathcal{C}/W \to \mathcal{C}/W$ defined in the internal language by letting

$$F \Big(X_{(a,t)} \mid (a,t) \in W \Big) =_{\mathrm{df}} \Bigg(\sum_{b \in B_a} X_{t(b)}^{C_b} \mid (a,t) \in W \Bigg) ,$$

is a dependent polynomial functor. Indeed, it is naturally isomorphic to the functor associated to the diagram

$$\begin{tikzcd}
\sum_{(a,t)\in W} \sum_{b \in B_a} C_b \arrow[rr, "p"] \arrow[dr, "s"'] & & \sum_{(a,t) \in W} B_a \arrow[dl, "r"] \\
& W &
\end{tikzcd}$$

where $p(a,t,b,c) =_{\mathrm{df}} (a,t,b)$, $s(a,t,b,c) =_{\mathrm{df}} t(b)$ and $r(a,t,b) =_{\mathrm{df}} (a,t)$ for $(a,t) \in W$, $b \in B_a$, $c \in C_b$.

(iii) If $\mathcal{C}$ has finite disjoint coproducts, the coproduct of two dependent polynomial functors is still a dependent polynomial functor. For two dependent polynomial functors $\mathcal{D}_1, \mathcal{D}_2$ associated respectively to the two diagrams

$$\begin{tikzcd}
B_1 \arrow[rr, "f_1"] \arrow[dr, "s_1"'] & & A_1 \arrow[dl, "r_1"] & B_2 \arrow[rr, "f_2"] \arrow[dr, "s_2"'] & & A_2 \arrow[dl, "r_2"] \\
& I & & & I &
\end{tikzcd}$$

the functor $\mathcal{D}_1 + \mathcal{D}_2$ is naturally isomorphic to the dependent polynomial functor associated to the diagram

$$\begin{tikzcd}
B_1 + B_2 \arrow[rr, "f_1 + f_2"] \arrow[dr, "{[s_1,s_2]}"'] & & A_1 + A_2 \arrow[dl, "{[r_1,r_2]}"] \\
& I &
\end{tikzcd}$$

5.1 Initial Algebras

We want to prove our first main result. We assume that the lccc $\mathcal{C}$ has $\mathcal{W}$-types.

Theorem 12. *Every dependent polynomial functor has an initial algebra.*

The proof involves a generalisation of the argument showing that $\mathcal{W}$-types exist in slice categories [16, Proposition 3.8], a result that follows indeed as a corollary of our theorem. We begin by constructing a candidate to be the initial algebra for the dependent polynomial functor defined in (3). Let us consider the $\mathcal{W}$-types $\mathcal{W}_f$ and $\mathcal{W}_{f\times I}$ associated to $f : B \to A$ and $f \times 1_I : B \times I \to A \times I$. The canonical isomorphisms

$$\mathcal{W}_f \cong \sum_{a\in A} \mathcal{W}_f^{B_a}\,, \qquad \mathcal{W}_{f\times I} \cong I \times \sum_{a\in A} \mathcal{W}_{f\times I}^{B_a}$$

will be treated here as equalities to simplify the presentation. Let us recall that there is an arrow $\rho : \mathcal{W}_f \to A$ defined by letting $\rho(a,t) =_{\mathrm{df}} a$ for $(a,t) \in \mathcal{W}_f$.

The strategy to define the candidate $V \to I$ to be an initial algebra will be as follows. First, we will define V as the object fitting in the equalizer diagram

$$\begin{tikzcd} V \arrow[r, "\eta"] & \mathcal{W}_f \arrow[r, shift left, "\xi"] \arrow[r, shift right, "\xi'"'] & \mathcal{W}_{f\times I} \end{tikzcd} \tag{4}$$

determined by appropriate arrows ξ and ξ'. Secondly, the required object of $\mathcal{C}/I$ can then be defined as $r\,\rho\,\eta : V \to I$. It now remains to define the arrows ξ, ξ'. The arrow ξ is defined by recursion on $\mathcal{W}_f$ by letting, for $(a,t) \in \mathcal{W}_f$,

$$\xi(a,t) =_{\mathrm{df}} (ra, a, (\lambda b \in B_a)\,\xi(tb))\,.$$

The definition of ξ' is more involved. First, we define $\phi : \mathcal{W}_{f\times I} \times B \to \mathcal{W}_{f\times I}$ by recursion. For $(i,a,t,b) \in \mathcal{W}_{f\times I}$, define

$$\phi(i,a,t,b) =_{\mathrm{df}} \big(sb, a, (\lambda b' \in B_a)\,\phi(t(b'), b')\big)\,.$$

Then, we define $\psi : \mathcal{W}_{f\times I} \to \mathcal{W}_{f\times I}$ by letting, for $(i,a,t) \in \mathcal{W}_{f\times I}$,

$$\psi(i,a,t) =_{\mathrm{df}} (i, a, t, (\lambda b \in B_a)\phi(tb, b))\,.$$

Finally, we fix $\xi' =_{\mathrm{df}} \psi\,\xi$. The key property of the object V that allows us to prove Theorem 12 is stated in the next lemma.

Lemma 13. *For all $(a,t) \in \mathcal{W}_f$, we have $(a,t) \in \sum_{a\in A_i} \prod_{b\in B_a} V_{sb}$ if and only if $(a,t) \in V_i$, where $i =_{\mathrm{df}} ra$.*

Proof. Let $(a,t) \in \mathcal{W}_f$ and define $i =_{\mathrm{df}} ra$. First of all one needs to show that, for all $b \in B_a$

$$\xi(tb) = \phi(\xi(tb), b) \iff sb = \rho(tb) \;\wedge\; \xi(tb) = \psi\,\xi(tb)\,. \tag{5}$$

This can be proved by unfolding the relevant definitions. We then get

$$\begin{aligned}(a,t) \in V_i &\iff \xi(a,t) = \xi'(a,t) && \text{by def. of } V\\ &\iff (\forall b \in B_a)\ \xi(tb) = \phi(\xi(tb), b) && \text{by def. of } \xi\,, \xi'\\ &\iff (\forall b \in B_a)\ sb = \rho(tb) \wedge \xi(tb) = \psi\ \xi(tb) && \text{by (5)}\\ &\iff (a,t) \in \textstyle\sum_{a \in A_i} \prod_{b \in B_a} V_{sb} && \text{by def. of } V\end{aligned}$$

as required. □

Lemma 13 shows that $V \to I$ can be equipped with a structure map, and thus gives us an algebra for the dependent polynomial functor defined in (3). The initiality of this algebra follows from reasoning that is completely analogous to that in [16, Proposition 3.8] and hence is omitted here.

5.2 Applications

We give a first application of Theorem 12. Let us consider two arrows $f : B \to A$ and $g : D \to C$ in a lccc $\mathcal{C}$ with $\mathcal{W}$-types. We can then define a bifunctor $F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ whose action on an object (X, Y) is defined by letting

$$F(X,Y) =_{\mathrm{df}} \mathcal{P}_f(X) \times \mathcal{P}_g(Y)\,.$$

For a fixed object X of $\mathcal{C}$, the functor $F^X : \mathcal{C} \to \mathcal{C}$ that maps Y into $F(X,Y)$ can easily be seen to be polynomial. It therefore has an initial algebra, that we denote as

$$F^X\big(\mu Y.F(X,Y)\big) \longrightarrow \mu Y.F(X,Y)$$

The assignment of $\mu Y.F(X,Y)$ to X can then be extended to a functor $\mathcal{C} \to \mathcal{C}$. We refer to these functors as *fixpoint functors.* We can now state our second main result.

Theorem 14. *Fixpoint functors are polynomial.*

Proof. We limit ourselves to sketch the main idea of the argument. Let us actually suppose that the fixpoint functor is polynomial, and let $Q \to P$ be an arrow in $\mathcal{C}$ such that

$$\mu Y.F(X,Y) \cong \sum_{p \in P} X^{Q_p}\,.$$

Direct calculations imply that there must be isomorphisms

$$P \cong A \times \sum_{c \in C} P^{D_c}\,,$$

$$Q_{(a,c,t)} \cong B_a + \sum_{d \in D_c} Q_{t(d)}$$

for $(a, c, t) \in A \times \sum_{c \in C} P^{D_c}$. The first isomorphism certainly holds if we define P as the $\mathcal{W}$-type of the arrow $g \times 1_A : D \times A \to C \times A$ and use Corollary 11. Theorem 12 shows that it is also possible to satisfy the second isomorphism by defining $Q \to P$ as the initial algebra for an appropriate dependent polynomial functor. Recalling the examples of dependent polynomial functors given earlier in this section, it is immediate to observe that the functor $F : \mathcal{C}/P \to \mathcal{C}/P$ defined by letting

$$F(X_{(a,c,t)} \mid (a, c, t) \in P) =_{\mathrm{df}} B_a + \sum_{d \in D_c} X_{t(d)}$$

for $(X_{(a,c,t)} \mid (a, c, t) \in P)$ in $\mathcal{C}/P$, is a dependent polynomial functors, since it is the sum of two such functors. □

6 Free Monads

6.1 Background

We review some facts concerning endofunctors and monads, and some results concerning free monads. More details can be found in [4,11].

Definition 15. Let P be an endofunctor on $\mathcal{C}$. We say that P *has a free monad* if the forgetful functor $U : P\text{-}alg \to \mathcal{C}$ has a left adjoint.

The next proposition shows that the existence of a free monad for an endofunctor is a necessary and sufficient condition for its category of algebras to be isomorphic to a category of algebras for a monad.

Proposition 16. *The forgetful functor $U : P\text{-}alg \to \mathcal{C}$ has a left adjoint if and only if it is monadic over $\mathcal{C}$.*

Proof. The proof is an application of Beck's theorem [13] characterising monadic adjunctions. One should observe that the functor U satisfies all the hypothesis of Beck's theorem except for the existence of a left adjoint. □

When (T, η, μ) is a monad on a category $\mathcal{C}$ we write $T\text{-}Alg$ for the usual category of T-algebras. Note that we follow a suggestion of Peter Freyd in using $P\text{-}alg$ for the algebras of an endofunctor P and $T\text{-}Alg$ for the algebras for a monad T. Again, we write $U : T\text{-}Alg \to \mathcal{C}$ for the forgetful functor. We can then restate Proposition 16 as follows.

Proposition 17. *(T, η, μ) is a free monad for P if and only if there is an equivalence $T\text{-}Alg \to P\text{-}alg$ such that the following diagram commutes*

$$\begin{tikzcd}
T\text{-}Alg \arrow[rr] \arrow[dr, "U"'] & & P\text{-}alg \arrow[dl, "U"] \\
& \mathcal{C} &
\end{tikzcd}$$

We wish to give a more concrete description of the free monad for an endofunctor on a locally cartesian closed category with coproducts. To do so, we use the family of functors $P_X : \mathcal{C} \to \mathcal{C}$, for X in $\mathcal{C}$, associated to a functor $P : \mathcal{C} \to \mathcal{C}$ and defined in Subsection 2.4. As we did in the discussion leading to Proposition 4, we assume that $\mathcal{C}$ has finite disjoint coproducts.

Proposition 18. *Let P be an endofunctor on $\mathcal{C}$. The following are equivalent:*

(i) the endofunctor P has a free monad;
(ii) the comma category $X \downarrow U$ has an initial object, for all X in $\mathcal{C}$;
(iii) the endofunctor P_X has an initial algebra, for all X in $\mathcal{C}$.

Proof. The equivalence (i) $\Leftrightarrow$ (ii) follows by Definition 15 and by the possibility of determining a left adjoint via initial objects in comma categories [13]. The equivalence (ii) $\Leftrightarrow$ (iii) follows from the isomorphism $X \downarrow U \cong P_X$*-alg*. One could also verify directly the implication (iii) $\Rightarrow$ (i) by defining explicitly the free monad (T, η, μ) for P. The functor T is defined by letting $T(X)$ be the initial algebra for P_X, for X in $\mathcal{C}$. □

We conclude this review by recalling the notion of strenth for a monad and a simple fact about it.

Definition 19. Let (T, η, μ) be a monad on $\mathcal{C}$. By a *strength* for (T, η, μ) we mean a strength σ for the functor T such that, for all X and Y in $\mathcal{C}$, we have

$$\sigma_{X,Y} \circ (1_X \otimes \eta_Y) = \eta_{X \otimes Y} \, , \quad \mu_{X \otimes Y} \circ T(\sigma_{X,Y}) \circ \sigma_{X,TY} = \sigma_{X,Y} \circ (1_X \otimes \mu_Y) \, .$$

Proposition 20. *Let P be an endofunctor on $\mathcal{C}$ and (T, η, μ) be the free monad on P. A strength for the functor P determines a strength for the monad (T, η, μ).*

Proof. The strength can be defined using the explicit description of the free monad given in the proof of Proposition 18. □

6.2 Free Monads for Polynomial Functors

We begin by ensuring the existence of free monads for polynomial functors.

Theorem 21. *If $\mathcal{C}$ is a lccc with finite disjoint coproducts and $\mathcal{W}$-types, then every polynomial endofunctor on $\mathcal{C}$ has a free monad.*

Proof. Let $P : \mathcal{C} \to \mathcal{C}$ be a polynomial functor. If we knew that for every X in $\mathcal{C}$ the functor $P_X : \mathcal{C} \to \mathcal{C}$ had an initial algebra, then we could invoke Proposition 18 and conclude the desired claim. By Proposition 4, however, the functors $P_X : \mathcal{C} \to \mathcal{C}$, for X in $\mathcal{C}$, are polynomial, and therefore they have an initial algebra by the assumption that $\mathcal{C}$ has $\mathcal{W}$-types. □

The next corollary, a consequence of Proposition 16 and Theorem 21, allows us to observe the existence of structure on the categories of algebras for polynomial functors. From now on, we assume that $\mathcal{C}$ is a lccc with finite disjoint coproducts and $\mathcal{W}$-types.

Corollary 22. *For every polynomial functor P on $\mathcal{C}$, the category P-alg is isomorphic to the category T-Alg, where T is free monad on P.*

We can also derive information on free monads for polynomial functors.

Proposition 23. *Free monads for polynomial functors have a strength.*

Proof. The claim is a consequence of Proposition 6 and Proposition 20. □

We conclude the paper with our third and last main result, whose proof is completely analogous to that of Theorem 14.

Theorem 24. *The free monad on a polynomial functor is polynomial.*

Acknowledgements. We thank Peter Aczel, Thorsten Altenkirch, Benno van den Berg, Marcelo Fiore and Erik Palmgren for stimulating discussions.

References

1. Abbott, M.: *Categories of Containers.* Ph.D. thesis, University of Leicester (2003).
2. Abbott, M., Altenkirch, T., Ghani, N.: Categories of Containers. in *Foundations of Software Science and Computation Structures* LNCS 2620, Springer (2003) 23 – 38.
3. Aczel, P.: The Type Theoretic Interpretation of Constructive Set Theory: Inductive Definitions, in: R.B. Barcan Marcus et al. (eds.) *Logic, Methodology and Philosophy of Science, VII*, North-Holland (1986).
4. Barr, M., Wells, C.: *Toposes, triples and theories.* Springer-Verlag (1985).
5. Dybjer, P: Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theoretical Computer Science* 176, (1997), 329 – 335.
6. Ehrhard, T., Regnier, L.: The differential lambda-calculus. *Theoretical Computer Science*, 309 (2003) 1 – 41.
7. Freyd, P.: Aspects of topoi. *Bull. Austral. Math. Soc.* 7 (1972) 1 – 76.
8. Griffor, E., Rathjen, M.: The Strength of Some Martin-Löf's Type Theories, *Archiv for Mathematical Logic* 33 (1994) 347 – 385.
9. Hofmann, M.: On the interpretation of type theory in locally cartesian closed categories. In *Computer Science Logic '94.* LNCS 933, Springer (1995) 427 – 441.
10. Kelly, G.M., Street, R.: Review of the elements of 2-categories. *Proc. Sydney Category Theory Seminar 1972/73* LNM 420, Springer (1974) 75 – 103.
11. Kelly, G.M.: A unified treatment of transfinite constructions for free algebras, free monoids, colimit, associated sheaves and so on. *Bull. Austral. Math. Soc.* 22 (1980) 1 – 83.
12. Kelly, G.M.: *Basic Concepts of Enriched Category Theory* Cambridge University Press (1982).
13. Mac Lane, S.: *Categories for the working mathematician.* Springer-Verlag (1998).
14. Maietti, M.E.: *The type theory of categorical universes.* Ph.D. thesis, Università di Padova (1998).
15. Martin-Löf, P. *Intuitionistic Type Theory.* Bibliopolis (1984).
16. Moerdijk, I., Palmgren, E.: Wellfounded trees in categories. *Annals of Pure and Applied Logic* 104 (2000) 189 – 218.

17. Nordström, B., Petersson K., Smith J.: *Programming in Martin-Löf Type Theory.* Oxford University Press (1990).
18. Seely, R.A.G.: Locally cartesian closed categories and type theory. *Math. Proc. Camb. Phil. Soc.* 95 (1984) 33 – 48.
19. Street, R.: The formal theory of monads. *J. of Pure and Appl. Algebra* 2 (1972) 149 – 168.