Representing Nested Inductive Types using W-types

Michael Abbott[1], Thorsten Altenkirch[2], and Neil Ghani[1]

[1] Department of Mathematics and Computer Science, University of Leicester
michael@araneidae.co.uk, ng13@mcs.le.ac.uk
[2] School of Computer Science and Information Technology, Nottingham University
txa@cs.nott.ac.uk

Abstract. We show that strictly positive inductive types, constructed from polynomial functors, constant exponentiation and arbitrarily nested inductive types exist in any Martin-Löf category (extensive locally cartesian closed category with W-types) by exploiting our work on container types. This generalises a result by Dybjer (1997) who showed that non-nested strictly positive inductive types can be represented using W-types. We also provide a detailed analysis of the categorical infrastructure needed to establish the result.

1 Introduction

Inductive types play a central role in programming and constructive reasoning. From an intuitionistic point of view we can understand strictly positive inductive types (SPITs) as well-founded trees, which may be infinitely branching. The language of SPITs is built from polynomial types and exponentials, enriched by a constructor μ for inductive types. In this language we can conveniently construct familiar types such as the natural numbers, $\mathbb{N} = \mu X.1 + X$; binary trees, $\mathrm{BTree} = \mu X.1 + X \times X$; lists parameterised over a type $\mathrm{List}\, A = \mu X.1 + A \times X$; ordinals, $\mathrm{Ord} = \mu X.1 + X + X^{\mathbb{N}}$; and finitely branching trees, $\mathrm{FTree} = \mu Y.\mathrm{List}\, Y = \mu Y.\mu X.1 + X \times Y$. Categorically, μ corresponds to taking the initial algebra of a given functor.

The grammar of SPITs can be easily defined inductively, see definition 7.1. However, we would like to have a simple semantic criterion which guarantees the existence of SPITs. Dybjer (1997) shows that inductive types over strictly positive operators constructed using only polynomials in a single type variable and fixed exponentiation can be constructed in extensional Type Theory using W-types, the type of well-founded trees introduced in Martin-Löf (1984). However, Dybjer (1997) does not allow any nesting of inductive types, e.g. the example FTree is not covered by his definition. Here we present a more general result which shows that nested inductive types can be constructed using only W-types and we analyse the categorical framework in more detail.

An important ingredient in our construction is the insight that parametrised SPITs give rise to containers, which we have investigated in Abbott et al. (2003) and which are the topic of Abbott (2003). The basic notion of a *container* is a dependent pair of types $A \vdash B$ creating a functor $T_{A \rhd B} X \equiv \sum a : A. X^{B(a)}$. A morphism of containers $(A \vdash$

$B) \to (C \vdash D)$ is a pair of morphisms $(u : A \to C,\ f : u^*D \to B)$. With this definition of a category $\mathscr{G}$ of containers we can construct a full and faithful functor $T : \mathscr{G} \to [\mathbb{C}, \mathbb{C}]$.

However, when constructing fixed points it is also necessary to take account of containers with parameters, so we define $T : \mathscr{G}_I \to [\mathbb{C}^I, \mathbb{C}]$ for each parameter index set I. For the purposes of this paper the index set I can be regarded as a finite set, but this makes little practical difference to the development.

It is easy to show that containers are closed under sums and products and constant exponentiation, see Abbott et al. (2003); this is also done in Dybjer, 1997 for containers in one variable. W-types are precisely the initial algebras of containers in one variable (theorem 4.3), hence constructing inductive types over a single variable SPITs is straightforward and already covered (in part) by Dybjer's work. However, the general case for nested types corresponds to showing that containers are closed under initial algebras. The problem boils down (proposition 5.1) to solving an equation on families of types up to isomorphism, which is achieved in proposition 6.1.

The work presented here also overcomes a shortcoming of Abbott et al. (2003): there we constructed initial algebras of containers using the assumption that the ambient category is locally finitely presentable. Alas, this assumption rules out many interesting examples of categories, in particular realisability models such as ω-sets. This is fixed here, since we only have to require that the category has all W-types, i.e. initial algebras of container functors, which can be easily established for realisability models. Since dependent types and inductive types are the core of Martin-Löf's Type Theory, we call categories with this structure *Martin-Löf categories*, see definition 4.4.

Dybjer and Setzer (1999, 2001) have a goal similar to ours in reducing schematic presentations of inductive types to simpler combinators. However, the focus of their work are inductive-recursive definitions such as universes which we do not consider here. Indeed, we manage to avoid having to introduce universes to carry out our constructions such as the isomorphism 6.1.

2 Definitions and Notation

This paper implicitly uses the machinery of fibrations (Jacobs 1999, Borceux 1994, chapter 8, etc) to develop the key properties of container categories, and in particular the fullness of the functor T relies on the use of fibred natural transformations. This section collects together the key definitions and results required in this paper.

Given a category with finite limits $\mathbb{C}$, refer to the slice category $\mathbb{C}/A$ over $A \in \mathbb{C}$ as the *fibre* of $\mathbb{C}$ over A. Pullbacks in $\mathbb{C}$ allow us to lift each $f : A \to B$ in $\mathbb{C}$ to a *pullback* or *reindexing* functor $f^* : \mathbb{C}/B \to \mathbb{C}/A$. Assigning a fibre category to each object of $\mathbb{C}$ and a reindexing functor to each morphism of $\mathbb{C}$ is (subject to certain coherence equations) a presentation of a *fibration over* $\mathbb{C}$.

Composition with f yields a functor $\sum_f : \mathbb{C}/A \to \mathbb{C}/B$ left adjoint to f^*. $\mathbb{C}$ is *locally cartesian closed* iff each fibre of $\mathbb{C}$ is cartesian closed, or equivalently, if each pullback functor f^* has a right adjoint $f^* \dashv \prod_f$. The exponential is written as B^A or $A \Rightarrow B$ when convenient.

Each exponential category $\mathbb{C}^I$ can in turn be regarded as fibred over $\mathbb{C}$ by taking the fibre of $\mathbb{C}^I$ over $A \in \mathbb{C}$ equal to $(\mathbb{C}/A)^I$. Now define $[\mathbb{C}^I, \mathbb{C}^J]$ to be the category of

fibred functors $F:\mathbb{C}^I \to \mathbb{C}^J$ and fibred natural transformations, where each F is a family $F_A:(\mathbb{C}/A)^I \to (\mathbb{C}/A)^J$ such that $(f^*)^J F_B \cong F_A(f^*)^I$ for each $f:A\to B$ and similarly for natural transformations.

Write $a:A \vdash B(a)$ or even just $A \vdash B$ for $B \in \mathbb{C}/A$. We'll write $A,B \vdash C$ or with variables $a:A, b:B(a) \vdash C(a,b)$ as a shorthand for $(a,b):\sum_A B \vdash C(a,b)$. Given $A \vdash B$ write $\pi_B:\sum_A B \to A$ for the corresponding *display map* for the type B. Write $\sum a:A$ and $\prod a:A$ for the $\sum$ and $\prod$ types corresponding to the adjoints to reindexing. Substitution in variables will be used interchangeably with substitution by pullback, so $A \vdash f^*B$ may also be written as $a:A \vdash B(f(a))$ or $a:A \vdash B(fa)$.

When dealing with a collection A_i for $i \in I$, we'll write this as $(A_i)_{i\in I}$ or $\vec{A}$ or even just A. The signs $\sum$ and $\prod$ will be used for both coproducts and products (respectively) over external sets and the corresponding internal constructions in $\mathbb{C}$. See Hofmann (1997) for a more detailed explanation of the interaction between type theory and semantics assumed in this paper.

The equality type $A,A \vdash \mathrm{Eq}_A$ is represented as an object of $\mathbb{C}/A\times A$ by the diagonal morphism $\delta_A:A\to A\times A$, and more generally $\Gamma,A,A \vdash \mathrm{Eq}_A$. Given parallel morphisms u,v into A the equality type has the key property that an element of $\mathrm{Eq}(u,v) = (u,v)^*\,\mathrm{Eq}_A$ exists precisely when $u=v$ as morphisms of $\mathbb{C}$.

Limits and colimits are *fibred* iff they exist in each fibre and are preserved by reindexing functors. Limits and colimits in a locally cartesian closed category $\mathbb{C}$ are automatically fibred. This useful result allows us to omit the qualification that limits and colimits be "fibred" throughout this paper.

When $\mathbb{C}$ is locally cartesian closed say that coproducts are *disjoint* (or equivalently that $\mathbb{C}$ is *extensive*)[1] iff the pullback of distinct coprojections $\kappa_i:A_i \to \sum_{i\in I} A_i$ into a coproduct is always the initial object 0. Henceforth, we'll assume that $\mathbb{C}$ has finite limits, is locally cartesian closed and has disjoint coproducts. For simplicity we call such a category an *extensive locally cartesian closed category*. The following notion of "disjoint fibres" follows from disjoint coproducts.

Proposition 2.1. *If $\mathbb{C}$ is locally cartesian closed and has disjoint coproducts then the functor $\vec{\kappa}^*:\mathbb{C}/\sum_{i\in I}A_i \to \prod_{i\in I}(\mathbb{C}/A_i)$, taking $\sum_{i\in I}A_i \vdash B$ to $(A_i \vdash \kappa_i^* B)_{i\in I}$, is an equivalence. Say that $\mathbb{C}$ has* disjoint fibres *when this holds.* □

Write $\biguplus:\prod_{i\in I}(\mathbb{C}/A_i) \to \mathbb{C}/\sum_{i\in I}A_i$ for the adjoint to $\vec{\kappa}^*$ and $-\dot{+}-$ for the binary case. Note that $\biguplus_{i\in I} B_i \cong \sum_{i\in I}\sum_{\kappa_i} B_i$ for $(A_i \vdash B_i)_{i\in I} \in \prod_{i\in I}(\mathbb{C}/A_i)$. For example, given $A \vdash B$ and $C \vdash D$ (with display maps π_B and π_D) we write $A+C \vdash B \dot{+} D$ for their disjoint sum; this satisfies two identities: $\sum_{A+C}(B\dot{+}D) \cong \sum_A B + \sum_C D$ and $\pi_{B\dot{+}D} = \pi_B + \pi_D$ (modulo the preceding isomorphism).

The following lemma collects together some useful identities which hold in any category considered in this paper.

[1] For general $\mathbb{C}$, coproducts are disjoint iff coprojections are also mono, and $\mathbb{C}$ is extensive iff coproducts are disjoint and are preserved by pullbacks.

Lemma 2.2. *For extensive locally cartesian closed* $\mathbb{C}$ *the following isomorphisms hold (*IC *stands for* intensional choice, Cu *for* Curry *and* DF *for* disjoint fibres*):*

$$\prod a:A.\ \sum b:B(a).\ C(a,b) \cong \sum f:\prod a:A.B(a).\ \prod a:A.\ C(a,fa) \qquad \text{(IC1)}$$

$$\prod_{i\in I} \sum b:B_i.\ C_i(b) \cong \sum a:\prod_{i\in I} B_i.\ \prod_{i\in I} C_i(\pi_i a) \qquad \text{(IC2)}$$

$$\prod a:A.\ C^{B(a)} \cong \left(\sum a:A.\ B(a)\right) \Rightarrow C \qquad \text{(Cu1)}$$

$$\prod_{i\in I} C^{B_i} \cong \left(\sum_{i\in I} B_i\right) \Rightarrow C \qquad \text{(Cu2)}$$

$$\left(\coprod_{i\in I} B_i\right)(\kappa_i a) \cong B_i(a) \qquad \text{(DF1)}$$

$$\sum_{i\in I} \sum a:A_i.\ C(\kappa_i a) \cong \sum a:\textstyle\sum_{i\in I} A_i.\ C(a) \qquad \text{(DF2)}\ \square$$

For technical convenience, a choice of pullbacks is assumed in $\mathbb{C}$.

Basic Properties of Containers

We summarise here the development of containers in Abbott et al. (2003).

Definition 2.3. *Given an index set* I *define the* category of containers $\mathscr{G}_I$ *as follows:*

- *Objects are pairs* $(A \in \mathbb{C},\ B \in (\mathbb{C}/A)^I)$*; write this as* $(A \triangleright B) \in \mathscr{G}_I$
- *A morphism* $(A \triangleright B) \to (C \triangleright D)$ *is a pair* (u, f) *for* $u : A \to C$ *in* $\mathbb{C}$ *and* $f : (u^*)^I D \to B$ *in* $(\mathbb{C}/A)^I$.

A container $(A \triangleright B) \in \mathscr{G}_I$ can be written using type theoretic notation as

$$\vdash A \qquad i:I, a:A \vdash B_i(a)\ .$$

A morphism $(u, f) : (A \triangleright B) \to (C \triangleright D)$ can be written in type theoretic notation as

$$u:A \longrightarrow C \qquad i:I, a:A \vdash f_i(a) : D_i(ua) \longrightarrow B_i(a)\ .$$

Finally, each $(A \triangleright B) \in \mathscr{G}_I$, thought of as a syntactic presentation of a datatype, generates a fibred functor $T_{A\triangleright B} : \mathbb{C}^I \to \mathbb{C}$ which is its semantics.

Definition 2.4. *Define the* container construction functor $T : \mathscr{G}_I \to [\mathbb{C}^I, \mathbb{C}]$ *as follows. Given* $(A \triangleright B) \in \mathscr{G}_I$ *and* $X \in \mathbb{C}^I$ *define*

$$T_{A\triangleright B}X \equiv \sum a:A.\ \prod_{i\in I} X_i^{B_i(a)}\ ,$$

and for $(u, f) : (A \triangleright B) \to (C \triangleright D)$ *define* $T_{u,f} : T_{A\triangleright B} \to T_{C\triangleright D}$ *to be the natural transformation* $T_{u,f}X : T_{A\triangleright B}X \to T_{C\triangleright D}X$ *thus:*

$$(a, g) : T_{A\triangleright B}X \vdash T_{u,f}X(a, g) \equiv (u(a), (g_i \cdot f_i)_{i\in I})\ .$$

The following proposition follows more or less immediately by the construction of T.

Proposition 2.5 (Abbott et al., 2003, proposition 3.3). *For each container $F \in \mathcal{G}_I$ and each container morphism $\alpha : F \to G$ the functor T_F and natural transformation T_α are fibred over $\mathbb{C}$.* □

By making essential use of the fact that the natural transformations in $[\mathbb{C}^I, \mathbb{C}]$ are fibred we can show that T is full and faithful.

Theorem 2.6 (ibid., theorem 3.4). *The functor $T : \mathcal{G}_I \to [\mathbb{C}^I, \mathbb{C}]$ is full and faithful.* □

This theorem gives a particularly simple analysis of polymorphic functions between container functors. For example, it is easy to observe that there are precisely n^m polymorphic functions $X^n \to X^m$: the data type X^n is the container $(1 \triangleright n)$ and hence there is a bijection between polymorphic functions $X^n \to X^m$ and functions $m \to n$. Similarly, any polymorphic function $\mathrm{List}X \to \mathrm{List}X$ can be uniquely written as a function $u : \mathbb{N} \to \mathbb{N}$ together with for each natural number $n : \mathbb{N}$ a function $f_n : un \to n$.

It turns out that each $\mathcal{G}_I$ inherits products and coproducts from $\mathbb{C}$, and that T preserves them:

Proposition 2.7 (ibid., propositions 4.1, 4.2). *If $\mathbb{C}$ has products and coproducts then $\mathcal{G}_I$ has products and coproducts preserved by T.* □

Given containers $F \in \mathcal{G}_{I+1}$ and $G \in \mathcal{G}_I$ we can compose their images under T to construct the functor

$$T_F[T_G] \equiv (\mathbb{C}^I \xrightarrow{(\mathrm{id}_{\mathbb{C}^I}, T_G)} \mathbb{C}^I \times \mathbb{C} \cong \mathbb{C}^{I+1} \xrightarrow{T_F} \mathbb{C}) \ .$$

This composition can be lifted to a functor $-[-] : \mathcal{G}_{I+1} \times \mathcal{G}_I \to \mathcal{G}_I$ as follows. For a container in $\mathcal{G}_{I+1}$ write $(A \triangleright B, E) \in \mathcal{G}_{I+1}$, where $B \in (\mathbb{C}/A)^I$ and $E \in \mathbb{C}/A$ and define:

$$(A \triangleright B, E)[(C \triangleright D)] \equiv \left(a : A,\ f : C^{E(a)} \triangleright \left(B_i(a) + \sum e : E(a).\ D_i(fe)\right)_{i \in I}\right) \ .$$

In other words, given type constructors $F(\vec{X}, Y)$ and $G(\vec{X})$ this construction defines the composite type constructor $F[G](\vec{X}) \equiv F(\vec{X}, G(\vec{X}))$.

Proposition 2.8 (ibid., proposition 6.1). *Composition of containers commutes with composition of functors thus: $T_F[T_G] \cong T_{F[G]}$.* □

This shows how composition of containers captures the composition of container functors.

3 Initial Algebras

In this section we discuss the construction of initial algebras for container functors and the principles in the ambient category $\mathbb{C}$ used to construct them.

Initial algebras can be regarded as the fundamental building blocks used to introduce recursive datatypes into type theory. Initial algebras define "well founded" structures, which can be regarded as the expression of terminating processes.

First some basic results about initial algebras.

Definition 3.1. *An* algebra *for a functor $F : \mathbb{C} \to \mathbb{C}$ is an object $X \in \mathbb{C}$ together with a morphism $h : FX \to X$; refer to X as the* carrier *of the algebra. An* algebra morphism *$(X,h) \to (Y,k)$ is a morphism $f : X \to Y$ satisfying the identity $f \cdot h = k \cdot Ff$. An* initial algebra *for F is then an initial object in the category of algebras and algebra morphisms.*

More explicitly, an initial algebra is an algebra $\alpha : FA \to A$ such that for any other algebra $h : FX \to X$ there exists a unique *morphism $\overline{h} : A \to X$ satisfying the equation $\overline{h} \cdot \alpha = h \cdot F\overline{h}$ thus:*

$$\begin{array}{ccc} FA & \xrightarrow{\alpha} & A \\ {\scriptstyle F\overline{h}}\downarrow & & \downarrow{\scriptstyle \overline{h}} \\ FX & \xrightarrow[h]{} & X \end{array} .$$

The following result tells us that initial algebras for a functor F are *fixed points* of F, and indeed the initial algebra is often called the least fixed point.

Proposition 3.2 (Lambek's Lemma). *Initial algebras are isomorphisms.* □

The following useful result about initial algebras tells us that initial algebras with parameters extend to functors.

Proposition 3.3. *Given a functor $F : \mathbb{D} \times \mathbb{C} \to \mathbb{C}$ if each endofunctor $F(X,-)$ on $\mathbb{C}$ has an initial algebra $(GX, \alpha X)$ then G extends to a functor and α to a natural transformation.* □

We can now define an operation μ constructing the least fixed point of a functors. If we regard a functor $F : \mathbb{D} \times \mathbb{C} \to \mathbb{C}$ as a type constructor $F(X,Y)$ then we can can regard the fixed points defined below as types.

Definition 3.4. *Given a functor $F : \mathbb{D} \times \mathbb{C} \to \mathbb{C}$ regarded as a type constructor $F(X,Y)$ define $\mu Y.F(X,Y)$ to be the initial algebra of the functor $F(X,-)$.*

To extend this definition of μ types to containers observe that for containers $F \in \mathcal{G}_{I+1}$ and $G \in \mathcal{G}_I$ the operation $G \mapsto F[G]$, with $T_{F[G]}X \cong T_F(X, T_G X)$ is an endofunctor on $\mathcal{G}_I$.

Definition 3.5. *For $F \in \mathcal{G}_{I+1}$ write μF for the initial algebra of $F[-] : \mathcal{G}_I \to \mathcal{G}_I$.*

We will show in this paper that the functor $\mu : \mathcal{G}_{I+1} \to \mathcal{G}_I$ exists, and that the initial algebra of a container functor is a container functor.

4 W-Types

In Martin-Löf's Type Theory (Martin-Löf, 1974; Nordström et al., 1990) the building block for inductive constructions is the W-type. Given a family of constructors $A \vdash B$ the type $\mathrm{W}a : A.B(a)$ (or $\mathrm{W}_A B$) should be regarded as the type of "well founded trees" constructed by regarding each $a : A$ as a constructor of arity $B(a)$.

The standard presentation of a W-type is through one type forming rule, an introduction rule and an elimination rule, together with an equation. As the type theoretic development in this paper focuses entirely on categorical models, we take W types to be *extensionally* defined, in particular the elimination rule constructs a *unique* term.

Definition 4.1. *A type system* has W-types *iff it has a type constructor*

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash \mathrm{W}_A B} \qquad \text{(W-type)}$$

together with a constructor term

$$\Gamma,\ a:A,\ f:(\mathrm{W}_A B)^{B(a)} \vdash \sup(a,b):\mathrm{W}_A B \qquad \text{(sup)}$$

and an elimination rule

$$\frac{\begin{array}{l}\Gamma,\ \mathrm{W}_A B \vdash C \\ \Gamma,\ a:A,\ f:(\mathrm{W}_A B)^{B(a)},\ g:\prod b:B(a).C(fb) \vdash h(a,f,g):C(\sup(a,f))\end{array}}{\Gamma,\ w:\mathrm{W}_A B \vdash \mathrm{wrec}_h(w):C(w)} \qquad \text{(wrec)}$$

satisfying the equation for variables $a:A$ and $f:(\mathrm{W}_A B)^{B(a)}$:

$$\mathrm{wrec}_h(\sup(a,f)) = h(a,f,\mathrm{wrec}_h \cdot f)\ .$$

Note that we can use the elimination rule together with equality types to conclude that wrec_h is unique.

Lemma 4.2. *Given variables $a:A$, $f:(\mathrm{W}_A B)^{B(a)}$ and $g:\prod_{\mathrm{W}_A B} C$ the following conditional equation holds:*

$$g(\sup(a,f)) = h(a,f,g\cdot f) \implies g = \mathrm{wrec}_h\ ,$$

in other words the term wrec_h *uniquely satisfies the induction equation.*

Proof. Given the equation $g(\sup(a,f)) = h(a,f,g\cdot f)$ it is sufficient to construct a term of type $\mathrm{Eq}(g(w),\mathrm{wrec}_h(w))$ for each $w:\mathrm{W}_A B$; inspecting the rule (wrec), it is enough to construct $\mathrm{Eq}(g(\sup(a,f)),\mathrm{wrec}_h(\sup(a,f)))$ from $\prod b:B(a).\mathrm{Eq}(g(fb),\mathrm{wrec}_h(fb))$, which is the same as $g\cdot f = \mathrm{wrec}_h\cdot f$. So calculate: $g(\sup(a,f)) = h(a,f,g\cdot f) = h(a,f,\mathrm{wrec}_h\cdot f) = \mathrm{wrec}_h(\sup(a,f))$. □

The W in W-type stands for "Well-ordering", and an element of $\mathrm{W}_A B$ can usefully be thought of as a well founded tree where each node of the tree is given by an element $a\in A$ and the elements $b\in B(a)$ represent the possible descendents of (or branches from) that node. Thus a tree can be described in two parts: a choice of element $a\in A$ together with a function f assigning to each $b\in B(a)$ a descendant tree to the branch b. Thus we get the constructor term $\sup: T_{A\triangleright B}(\mathrm{W}_A B)\to \mathrm{W}_A B$.

The elimination (or *induction*) rule (wrec) arises from the fact that every possible path through a tree $w\in \mathrm{W}_A B$ is finite. The constructor $h(a,f,g)$ constructs a new output over the tree $w=\sup(a,f)$ from the values available over all the descendants of w, namely fb for each $b\in B(a)$. The parameter g programs in the availability of these values.

W-types are initial algebras for a particularly familiar class of functors:

Theorem 4.3. W*-types are precisely the initial algebras of container functors in one parameter:*

$$\mathrm{W}_A B \cong \mu X.\, \textstyle\sum_A X^B = \mu X.\, T_{A \triangleright B} X \quad .$$

Proof. First observe that the map $\sup : T_{A \triangleright B} \mathrm{W}_A B \to \mathrm{W}_A B$ makes $\mathrm{W}_A B$ into an initial $T_{A \triangleright B}$-algebra: given an algebra $k : T_{A \triangleright B} X \to X$ define $h(a, f, g) \equiv k(a, g)$ and construct $\overline{k} \equiv \mathrm{wrec}_h$. To show that this is an algebra morphism, calculate in context $a : A$ and $f : (W_A B)^{B(a)}$

$$\overline{k} \sup(a, f) = \mathrm{wrec}_h(\sup(a, f)) = h(a, f, \mathrm{wrec}_h \cdot f) = k(a, \overline{k} \cdot f) = k(T_{A \triangleright B} \overline{k})(a, f)$$

and uniqueness of this map follows immediately: if a map $g : \mathrm{W}_A B \to X$ also satisfies $g \cdot \sup = k \cdot T_{A \triangleright B} g$, then in particular $g(\sup(a, f)) = k(a, g \cdot f) = h(a, f, g \cdot f)$ and so $g = \overline{k}$.

Conversely, to show that $\mu X. T_{A \triangleright B} X$ is a W-type we'll need to do a little more work. For conciseness write $Z \equiv \mu X. T_{A \triangleright B} X$ and $\sup : T_{A \triangleright B} Z \to Z$ for the initial algebra morphism on Z. Now let $Z \vdash C$ and h be given as in the hypotheses of the rule (wrec).

The initiality of Z will only allow us to construct a map into a constant type, so define $D \equiv \sum_Z C$ and from h construct $h' : T_{A \triangleright B} D \to D$ as follows.

First observe that $T_{A \triangleright B} D = \sum_A (\sum_Z C)^B \cong \sum_A \sum f : Z^B . \prod b : B . C(fb)$ (by intensional choice), and so we can write the arguments of h' as $a : A$, $f : Z^{B(a)}$ and $g : \prod b : B(a) . C(fb)$ and define $h'(a, f, g) = (\sup(a, f), h(a, f, g))$ as the following composite map:

$$T_{A \triangleright B} D = \sum_A \left(\sum_Z C\right)^B \cong \sum_A \sum f : Z^B .\, \prod b : B .\, C(fb) \xrightarrow{(\sup, h)} \sum_Z C \quad .$$

Initiality of sup induces $\overline{h} : Z \to \sum_Z C$ uniquely satisfying $\overline{h} \cdot \sup = h' \cdot T_{A \triangleright B} \overline{h}$. Write $\overline{h} = (\overline{h}_0, \overline{h}_1)$ and we can now write $T_{A \triangleright B} \overline{h} \cdot (a, f) = (a, \overline{h}_0 \cdot f, \overline{h}_1 \cdot f)$ as an element of $\sum_A \sum_{Z^B} \prod_B \varepsilon^* C$. This equation can now be written as a pair of equations

$$\overline{h}_0(\sup(a, f)) = \sup(a, \overline{h}_0 \cdot f) \quad , \qquad \overline{h}_1(\sup(a, f)) = h(a, \overline{h}_0 \cdot f, \overline{h}_1 \cdot f) \quad .$$

The equation for $\overline{h}_0$ tells us (by initiality of Z) that in fact $h_0 = \mathrm{id}_Z$, and then the second equation tells us that $\overline{h}_1 = \mathrm{wrec}_h$, and so $Z \cong \mathrm{W}_A B$ as required. □

We consider that this notion summarises the essence of Martin-Löf's Type Theory from a categorical perspective, hence the following definition.

Definition 4.4. *A* Martin-Löf category *is an extensive locally cartesian closed category with an initial algebra for every container functor (i.e. W-types).*

Constructing W-Types

We can either assume that $\mathbb{C}$ has W-types given axiomatically or, if $\mathbb{C}$ satisfies the necessary preconditions, derive them from theorem 4.8 below. Alternatively if $\mathbb{C}$ is a topos we can appeal to proposition 3.6 of Moerdijk and Palmgren (2000).

Proposition 4.5 (Moerdijk and Palmgren, 2000, proposition 3.6). *W-types exist in any elementary topos with a natural numbers object.* □

In the rest of this section we'll construct W-types from colimits in $\mathbb{C}$. First we need to set up some auxiliary machinery, deriving largely from Adámek and Rosický (1994). In particular, the result in this subsection as stated here relies on classical set-theoretic reasoning.

Recall that a category $\mathbb{J}$ is said to be *filtered* iff every finite and non-empty diagram in $\mathbb{J}$ has a compatible cocone in $\mathbb{J}$. In Adámek and Rosický (1994) we have the following extension of this notion to an arbitrary regular cardinal $\aleph$.

Definition 4.6. *Say that a category $\mathbb{J}$ is* $\aleph$-filtered *iff every subcategory of $\mathbb{J}$ with less than $\aleph$ morphisms has a compatible cocone.*

If a functor $F : \mathbb{C} \to \mathbb{C}$ preserves all $\aleph$-filtered colimits say that F has rank $\aleph$*, and say that F* has rank *iff it has rank for some $\aleph$.*

Note that an ordinary filtered category is precisely an $\aleph_0$-filtered category.

We now have the following important folklore result (see Abbott et al., 2003, theorems 5.6, 5.7). A variant of this theorem is proved in Adámek and Koubek (1979), and the case for $\aleph_0$ is a standard result in computer science (eg, Poigné, 1992, §7.3).

Proposition 4.7. *If $\mathbb{C}$ has an initial object and colimits of all filtered diagrams then any functor $F : \mathbb{C} \to \mathbb{C}$ with rank has an initial algebra.*

Proof. A sketch proof follows. In the finite case ($\aleph = \aleph_0$) we construct the colimit of the ω-chain

$$0 \longrightarrow F0 \longrightarrow F^2 0 \longrightarrow \cdots \longrightarrow \varinjlim_{n \in \aleph_0} F^n 0 \ .$$

Since F preserves this diagram, we can compute $F \varinjlim_n F^n 0 \cong \varinjlim_n F F^n 0 \cong \varinjlim_n F^n 0$; it is a straightforward calculation to verify that this is the required initial algebra morphism.

The generalisation to arbitrary $\aleph$ is a not altogether straightforward set theoretic generalisation of this result. □

We now obtain the following result.

Theorem 4.8. *If $\mathbb{C}$ is locally cartesian closed and locally presentable then $\mathbb{C}$ has all* W*-types.*

Proof. It will suffice to show that every one-parameter container functor $T_{A \triangleright B} : \mathbb{C} \to \mathbb{C}$ has rank, and hence has an initial algebra. Decompose $T_{A \triangleright B}$ into the chain of functors

$$\mathbb{C} \xrightarrow{A^*} \mathbb{C}/A \xrightarrow{(-)^B} \mathbb{C}/A \xrightarrow{\Sigma_A} \mathbb{C} \ .$$

We appeal to two results of Adámek and Rosický (1994) to show that all of these functors (and hence their composite) have rank. We know from their theorem 2.39 that each $\mathbb{C}/A$ is accessible, and their proposition 2.23 tells us that every functor between accessible categories with an adjoint has rank. □

5 Initial Algebras of Containers

One consequence of theorem 4.3 is that in the presence of W-types we can immediately construct μ types for containers in one parameter. However, the construction of a μ type for a container in multiple parameters is a more delicate matter and will require the introduction of more machinery.

Let $F : \mathbb{C}^{I+1} \to \mathbb{C}$ be a container in multiple parameters, which we can write as

$$F(X,Y) \equiv T_{S \triangleright P,Q}(X,Y) = \sum s{:}S.\ \left(\prod i{:}I.\ X_i^{P_i(s)}\right) \times Y^{Q(s)} = \sum_S \left(\prod_I X^P \times Y^Q\right) \ .$$

The task is to compute $(A \triangleright B)$ such that $T_{A \triangleright B} X \cong \mu Y.F(X,Y)$. Clearly

$$A \cong T_{A \triangleright B} 1 \cong \mu Y.\ F(1,Y) \cong \mu Y.\ \sum s{:}S.\ Y^{Q(s)} \cong \mathrm{W}_S Q \ ,$$

but the construction of $\mathrm{W}_S Q \vdash B$ is more tricky.

In the rest of this paper we will ignore the index set I and write X^P for $\prod_I X^P$. In particular, this means that the family $B \in (\mathbb{C}/\mathrm{W}_S Q)^I$ will be treated uniformly (as if $I = 1$). The required extra working to take account of I can be routinely added, but will further complicate a presentation which is quite complex enough already. We will therefore take

$$F(X,Y) \equiv \sum_S (X^P \times Y^Q) \ .$$

To simplify the algebra of types we will write $S, A^Q \vdash P + \sum_Q \varepsilon^* B$ as an abbreviation for the type expression (where ε is an evaluation map $A^Q \times Q \to A$):

$$s{:}S,\ f{:}A^{Q(s)} \ \vdash\ P(s) + \sum q{:}Q(s).\ B(fq) \ .$$

For conciseness write the initial algebra on $A = \mathrm{W}_S Q$ as $\psi : \sum_S A^Q \to A$.

Proposition 5.1. *Given the notation above, if $\mathrm{W}_S Q \vdash B$ is equipped with an isomorphism*

$$S, A^Q \ \vdash\ \varphi : P + \sum_Q \varepsilon^* B \cong \psi^* B$$

then $T_{A \triangleright B} X \cong \mu Y.F(X,Y)$.

Proof. First we show that each $T_{A \triangleright B} X$ is an $F(X,-)$ algebra thus:

$$\begin{aligned} F(X, T_{A \triangleright B} X) &= \sum_S \left(X^P \times \left(\sum_A X^B\right)^Q\right) \cong \sum_S \left(X^P \times \sum_{A^Q} \prod_Q X^{\varepsilon^* B}\right) \\ &\cong \sum_S \sum_{A^Q} \left(X^P \times \prod_Q X^{\varepsilon^* B}\right) \cong \sum_S \sum_{A^Q} X^{P + \sum_Q \varepsilon^* B} \\ &\overset{\varphi^{-1}}{\cong} \sum_S \sum_{A^Q} X^{\psi^* B} \overset{(\psi, \mathrm{id})}{\cong} \sum_A X^B = T_{A \triangleright B} X \ . \end{aligned}$$

With variables $s : S$, $g : X^{P(s)}$ and $h : \left(\sum_A X^B\right)^{Q(s)}$ note that we can decompose h into components $\pi \cdot h : A^{Q(s)}$ and $\pi' \cdot h : \prod q{:}Q(s).X^{B(\pi h q)}$ and so the algebra morphism $in : F(X, T_{A \triangleright B} X) \to T_{A \triangleright B} X$ can be conveniently written as

$$in(s,g,h) = \left(\psi(s, \pi \cdot h), \left[g; \pi' \cdot h\right] \cdot \varphi^{-1}\right) \ ;$$

conversely, given variables $s:S$, $f:A^{Q(s)}$ and $k:X^{B(\psi(s,f))}$ similarly note that $k\cdot\varphi\cdot\kappa'$ can be regarded as a term of type $\prod q:Q(s).X^{B(fq)}$ and so we can write

$$in^{-1}(\psi(s,f),k) = (s,\ k\cdot\varphi\cdot\kappa,\ (f,k\cdot\varphi\cdot\kappa'))\ .$$

To show that *in* is an *initial* $F(X,-)$-algebra we need to construct from any algebra $\alpha:F(X,Y)\to Y$ a unique map $\overline{\alpha}:T_{A\triangleright B}X\to Y$ satisfying the algebra morphism equation $\overline{\alpha}\cdot in = \alpha\cdot F(X,\overline{\alpha})$:

$$\begin{array}{ccc} F(X,T_{A\triangleright B}X) & \xrightarrow{\ in\ } & T_{A\triangleright B}X \\ {\scriptstyle F(X,\overline{\alpha})}\downarrow & & \downarrow{\scriptstyle \overline{\alpha}} \\ F(X,Y) & \xrightarrow[\ \alpha\ ]{} & Y \end{array}\ .$$

The map $\overline{\alpha}$ can be transposed to a term $A\vdash\widetilde{\alpha}:X^B\Rightarrow Y$ which we will construct by induction on $A=\mathrm{W}_S Q$. Given $s:S$, $f:A^{Q(s)}$ and $k:X^{B(\psi(s,f))}$ construct $g\equiv k\cdot\varphi\cdot\kappa:X^{P(s)}$ and $h\equiv k\cdot\varphi\cdot\kappa':\prod q:Q(s).X^{B(fq)}$. In this context define $H(s,f,\beta)(k)\equiv\alpha(s,g,\beta(h))$ and compute

$$\begin{aligned}\widetilde{\alpha}(\psi(s,f))(k) &= \overline{\alpha}(\psi(s,f),k) = \overline{\alpha}\cdot in\cdot(s,g,(f,h)) \\ &= \alpha\cdot F(X,\overline{\alpha})\cdot(s,g,(f,h)) = \alpha(s,g,\overline{\alpha}\cdot(f,h)) \\ &= \alpha(s,g,(\widetilde{\alpha}\cdot f)(h)) = H(s,f,\widetilde{\alpha}\cdot f)(k)\ .\end{aligned}$$

This shows that $\widetilde{\alpha}=\mathrm{wrec}_H$ and thus that $T_{A\triangleright B}X$ is an $F(X,-)$-initial algebra. □

Note that as a corollary of this proposition the isomorphism $P+\sum_Q\varepsilon^*B\cong\psi^*B$ over $\mathrm{W}_S Q$ defines B up to isomorphism, since the container $T_{A\triangleright B}$ is determined up to isomorphism as an initial algebra.

Of course, it remains to prove the hypothesis of the theorem above, that a family $A\vdash B$ with the given isomorphism φ exists; we do this in proposition 6.1.

6 Constructing a Fixed Point over an Initial Algebra

Proposition 5.1 relies on the hypothesis that the functor $X\mapsto P+\sum_Q\varepsilon^*X$ has a fixed point "over" the initial algebra $\psi:T_{S\triangleright Q}A\to A$, or in other words there exists a B such that $P+\sum_Q\varepsilon^*B\cong\psi^*B$. This fixed point does indeed exist, as a *subtype* of a W-type.

Proposition 6.1. *For each fixed point $\psi:T_{S\triangleright Q}A\cong A$ there exists an object $A\vdash B$ such that there is an isomorphism:*

$$S,A^Q\ \vdash\ P+\sum\nolimits_Q\varepsilon^*B\cong\psi^*B\ .$$

Proof. Write $S,A^Q\vdash\varphi:P+\sum_Q\varepsilon^*B\to\psi^*B$ for the isomorphism that we wish to construct. As already noted, we cannot directly appeal to W-types to construct this fixed

point, so the first step is to create a fixed point equation that we *can* solve. Begin by "erasing" the type dependency of B and construct (writing $\sum_Q Y \cong Q \times Y$, etc)

$$\begin{aligned}\widehat{B} &\equiv \mu Y.\ \textstyle\sum_S \sum_{A^Q} \big(P + Q \times Y\big) \cong \mu Y.\ \Big(\sum_S (A^Q \times P) + \big(\sum_S (A^Q \times Q)\big) \times Y\Big) \\ &\cong \mathrm{List}\Big(\textstyle\sum_S (A^Q \times Q)\Big) \times \sum_S (A^Q \times P)\ ;\end{aligned}$$

there is no problem in constructing arbitrary lists in $\mathbb{C}$ so $\widehat{B}$ clearly exists.

The task now is to select the "well-formed" elements of $\widehat{B}$. A list in $\widehat{B}$ can be thought of as a putative path through a tree in $\mu Y.\, T_{S \triangleright P,Q}(X,Y)$; we want $B(a)$ to be the set of all valid paths to X-substitutable locations in the tree.

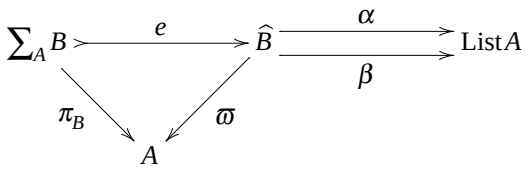
An element of $\widehat{B}$ can be conveniently written as a list followed by a tuple thus

$$([(s_0, f_0, q_0), \ldots, (s_{n-1}, f_{n-1}, q_{n-1})], (s_n, f_n, p))$$

for $s_i : S$, $f_i : A^{Q(s_i)}$, $q_i : Q(s_i)$ and $p : P(s_n)$. The condition that this is a well formed element of $B(\psi(s_0, f_0))$ can be expressed as the n equations

$$f_i(q_i) = \psi(s_{i+1}, f_{i+1}) \quad \text{for } i < n$$

which can be captured as an equaliser diagram



where α, β and ϖ are defined inductively on $\widehat{B}$ as follows (and $\pi_B \equiv \varpi \cdot e$):

$$\begin{aligned} \alpha(\mathsf{nil}, p') &= \mathsf{nil} & \alpha(\mathsf{cons}((s,f,q),l), p') &= \mathsf{cons}(fq, \alpha(l,p')) \\ \varpi(\mathsf{nil}, (s,f,p)) &= \psi(s,f) & \varpi(\mathsf{cons}((s,f,q),l), p') &= \psi(s,f) \\ \beta(\mathsf{nil}, p') &= \mathsf{nil} & \beta(\mathsf{cons}(b,l), p') &= \mathsf{cons}(\varpi(l,p'), \beta(l,p'))\ . \end{aligned}$$

The property that $b : \widehat{B}$ is an element of B can be written $b : B(\varpi b)$ and can be expressed inductively as follows:

$$\top \implies (\mathsf{nil}, (s,f,p)) : B(\psi(s,f)) \tag{1}$$

$$fq = \varpi(l,p') \wedge (l,p') : B(fq) \implies (\mathsf{cons}((s,f,q),l), p') : B(\psi(s,f))\ . \tag{2}$$

The converse to (2) also holds, since $(\mathsf{cons}((s,f,q),l),p') : B(\psi(s,f)) \iff \mathsf{cons}(fq, \alpha(l,p')) = \mathsf{cons}(\varpi(l,p'), \beta(l,p')) \iff fq = \varpi(l,p') \wedge (l,p') : B(fq)$.

The isomorphism $\widehat{\varphi} : \sum_S \sum_{A^Q} (P + Q \times \widehat{B}) \cong \widehat{B}$ can now be used to construct the isomorphism φ for B. Writing an element of $\sum_S \sum_{A^Q} (P + Q \times \widehat{B})$ as $(s, f, \kappa p)$ or $(s, f, \kappa'(q,b))$, the function $\widehat{\varphi}$ can be computed thus:

$$\begin{array}{ccc} \sum_S \sum_{A^Q} (P + Q \times \widehat{B}) & \overset{\widehat{\varphi}}{\cong} & \mathrm{List}\big(\sum_S (A^Q \times Q)\big) \\ & & \times \sum_S (A^Q \times P) \end{array} = \widehat{B}$$

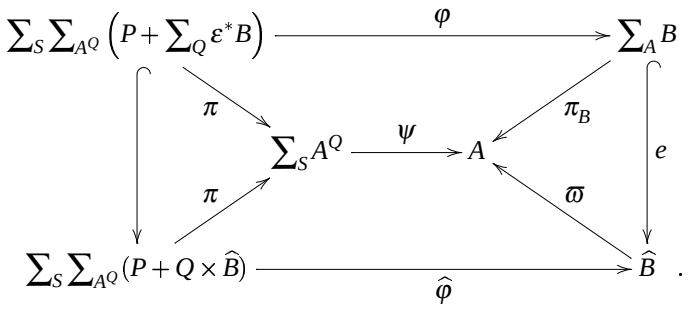
$$\begin{aligned} (s, f, \kappa p) &\longleftrightarrow (\mathsf{nil}, (s,f,p)) \\ (s, f, \kappa'(q, (l,p'))) &\longleftrightarrow (\mathsf{cons}((s,f,q),l), p')\ . \end{aligned}$$

To show that $\widehat{\varphi}$ restricts to a morphism $\varphi : P + \sum_Q \varepsilon^* B \to \psi^* B$ we need to show for each $s : S$ and $f : A^Q$ that $x : (P(s) + \sum q : Q(s).\, B(fq))$ implies $\widehat{\varphi}(s, f, x) : B(\psi(s, f))$.

When $x = \kappa p$ we immediately have $\widehat{\varphi}(s, f, \kappa p) = (\mathsf{nil}, (s, f, p)) : B(\psi(s, f))$ by (1) above. Now let $(s, f, \kappa'(q, (l, p')))$ be given with $(l, p') : B(fq)$ (which means, in particular, that $\varpi(l, p') = fq$) and consider the equation $\widehat{\varphi}(s, f, \kappa'(q, (l, p'))) = (\mathsf{cons}((s, f, q), l), p')$, then by (2) this is also in $B(\psi(s, f))$. Thus $\widehat{\varphi}$ restricts to

$$s : S, f : A^{Q(s)} \;\vdash\; \varphi_{s,f} : P(s) + \sum q : Q(s).\; B(fq) \longrightarrow B(\psi(s, f)) \quad .$$

We have in effect constructed φ making the diagram below commute:



To show that φ is an isomorphism we need to show that $\widehat{\varphi}^{-1}$ restricts to an inverse to φ. As before we can analyse $b : B(\psi(s, f))$ into two cases, and show that in both cases $\widehat{\varphi}^{-1} b : P(s) + \sum q : Q(s).\, B(fq)$.

When $b = (\mathsf{nil}, (s, f, p))$ then $\widehat{\varphi}^{-1} b = (s, f, \kappa p)$ which can be regarded as an element of $P(s)$. When $b = (\mathsf{cons}((s, f, q), l), p')$ and so $\widehat{\varphi}^{-1} b = (s, f, \kappa'(q, (l, p')))$ it is enough to observe that $b : B(\psi(s, f))$ implies $(l, p') : B(fq)$ and hence $\widehat{\varphi}^{-1} b$ arises from an element of $\sum q : Q(s).\, B(fq)$. □

We conclude our development with the following summary result as a corollary.

Corollary 6.2. *If $\mathbb{C}$ has W-types then containers are closed under the construction of μ-types.*

Note that that since μF is a fixed point, it satisfies the isomorphism $\mu F \cong F[\mu F]$.

7 Strictly Positive Inductive Types

We now have enough machinery in place to observe that all strictly positive types can be described as containers.

Definition 7.1. *A* strictly positive inductive type (SPIT) in n variables *(Abel and Altenkirch, 2000) is a type expression (with type variables $X_1, \ldots, X_n$) built up inductively according to the following rules:*

- *if K is a constant type (with no type variables) then K is a SPIT;*
- *each type variable X_i is a SPIT;*

– *if F, G are SPITs then so are $F+G$ and $F\times G$;*
– *if K is a constant type and F a SPIT then $K\Rightarrow F$ is a SPIT;*
– *if F is a SPIT in $n+1$ variables then $\mu X.F$ is a SPIT in n variables (for X any type variable).*

Note that the type expression for a SPIT F can be interpreted as a functor $F:\mathbb{C}^n\to\mathbb{C}$, and indeed we can see that each strictly positive type corresponds to a container in $\mathscr{G}_n$.

Let strictly positive types F, G be represented by containers $(A\triangleright B)$ and $(C\triangleright D)$ respectively, then the table below shows the correspondence between strictly positive types and containers.

$$\begin{array}{ll} K\mapsto (K\ \triangleright\ 0) & X_i\mapsto (1\ \triangleright\ (\delta_{i,j})_{j\in I}) \\ F+G\mapsto (A+C\ \triangleright\ B\dot{+}D) & F\times G\mapsto (a{:}A,\ c{:}C\ \triangleright\ B(a)\times D(c)) \\ K\Rightarrow F\mapsto \left(f{:}A^K\ \triangleright\ \sum k{:}K.\ B(fk)\right) & \end{array}$$

As we have seen in this paper the construction of fixed points can be described in a uniform way. Let F be represented by $(S\triangleright P,Q)\in\mathscr{G}_{I+1}$, then for each fixed point $\psi: T_{S\triangleright Q}A\cong A$ of $T_{S\triangleright Q}$ we have constructed in proposition 6.1 an isomorphism over ψ, written here as $A\vdash B_A$, of the form

$$s{:}S,\ f{:}A^{Q(s)}\ \vdash\ \varphi{:}P(s)+\sum q{:}Q(s).\ B_A(fs)\longrightarrow B_A(\psi(s,f))\ ;$$

we can now define

$$\mu Y.\ F\mapsto (\mathrm{W}_S Q\ \triangleright\ B_{\mathrm{W}_S Q})\ .$$

8 Discussion and further work

An important extension of the work presented here is to include coinductive types (ν), corresponding to terminal coalgebras, to cover non-well founded data structures such as streams ($\mathrm{Stream}\,A=\nu X.A\times X$), which are used extensively in lazy functional programming. We conjecture (see Abbott, 2003, p. 78), that Martin-Löf categories are closed under ν-types — this can be reduced to constructing the dual of W-types which we dub M-types.

Another interesting extension would be to consider inductive and coinductively defined families (such as vectors or simply typed λ-terms). Again, we conjecture that it should be possible to represent those within Martin-Löf categories. This result would provide further evidence establishing that these categories provide a convenient and concise base for intuitionistic Type Theory.

References

M. Abbott. *Categories of Containers*. PhD thesis, University of Leicester, 2003.

M. Abbott, T. Altenkirch, and N. Ghani. Categories of containers. In *Proceedings of Foundations of Software Science and Computation Structures*, volume 2620 of *Lecture Notes in Computer Science*, 2003.

A. Abel and T. Altenkirch. A predicative strong normalisation proof for a λ-calculus with interleaving inductive types. In *Types for Proof and Programs, TYPES '99*, volume 1956 of *Lecture Notes in Computer Science*, 2000.

J. Adámek and V. Koubek. Least fixed point of a functor. *Journal of Computer and System Sciences*, 19:163–178, 1979.

J. Adámek and J. Rosický. *Locally Presentable and Accessible Categories*. Number 189 in London Mathematical Society Lecture Note Series. Cambridge University Press, 1994.

F. Borceux. *Handbook of Categorical Algebra 2*, volume 51 of *Encyclopedia of Mathematics*. Cambridge University Press, 1994.

P. Dybjer. Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theoretical Computer Science*, 176:329–335, 1997.

P. Dybjer and A. Setzer. A finite axiomatization of inductive-recursive definitions. In *Typed Lambda Calculus and Applications*, pages 129–146, 1999.

P. Dybjer and A. Setzer. Indexed induction-recursion. *Lecture Notes in Computer Science*, 2183, 2001.

M. Hofmann. Syntax and semantics of dependent types. In A. M. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, volume 14, pages 79–130. Cambridge University Press, 1997.

B. Jacobs. *Categorical Logic and Type Theory*. Number 141 in Studies in Logic and the Foundations of Mathematics. Elsevier, 1999.

P. Martin-Löf. An intuitionistic theory of types: Predicative part. In H. E. Rose and J. C. Shepherdson, editors, *Proceedings of the Logic Colloquium*, pages 73–118. North-Holland, 1974.

P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Napoli, 1984.

I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Annals of Pure and Applied Logic*, 104:189–218, 2000.

B. Nordström, K. Petersson, and J. M. Smith. *Programming in Martin-Löf's Type Theory*. Number 7 in International Series of Monographs on Computer Science. Oxford University Press, 1990.

A. Poigné. Basic category theory. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1 of *Handbook of Logic in Computer Science*. Oxford University Press, 1992.