Recursive Models of General Inductive Types*

Yuxi Fu†

Department of Computer Science, University of Manchester
Manchester M13 9PL, England

Abstract. We give an interpretation of Martin-Löf's type theory (with universes) extended with generalized inductive types. The model is an extension of the recursive model given by Beeson. By restricting our attention to PER model, we show that the strictness of positivity condition in the definition of generalized inductive types can be dropped. It therefore gives an interpretation of general inductive types in Martin-Löf's type theory.

1 Introduction

Interest in inductively defined types has been around for some time. An early work on a categorical approach to recursively defined types is in [31], where the authors demonstrate that the initial T-algebra approach is a good alternative to what was proposed by ADJ-group. Later on, several researchers have investigated the idea of extending the existing typed calculi with inductively (coinductively) defined types. In [21, 22, 24], see also [7], the author considers a particular version of inductive and coinductive types. Two combinators and two rules are added to the second order λ-calculus to capture recursion and computational rules. In contrast to what is studied in other works, the notion of subtypes is used. That might cause some inconvenience; but it does enjoy a certain degree of conceptual clarity if one has in mind a constructive set-theoretical semantics. A general scheme for adding inductively defined types in Martin-Löf's type theory and in *ECC* are considered in [3, 4, 10, 11] and in [8, 20]. In both cases, the motivating example is the well-known W-types. The basic idea is to formalize the introductions of, and inductions on, data types. This method is more intuitive, and akin to the type theoretical tradition, than the initial T-algebra approach. The only primitive types assumed here are the Π-types with intensional equality, as opposed to the initial T-algebra case where one wants in addition the disjoint sums and the unit type at least.

A different methodology is to code up all inductive types in an extensional type theory with enough type constructors. This is adopted in [32, 30, 16, 14]. The argument against this approach is threefold. First, encoded inductive types are sometimes not good enough. For instance, the polymorphic encoding of the natural numbers type verifies only a weak form of recursion. Second, the meta-theory of the extensional theory (with the η-rule and the surjective paring rule) is hard. As a matter of fact, useful results in this area are almost non-existent. There are many conjectures though! Third, the encodings of inductive types via W-types look messy. Those reasons, together with the result in [9] which shows that the encodings are impossible in an intensional type theory, are convincing enough for us to prefer the traditional method.

The formulation of inductive types as initial T-algebras is studied in [29]. The construction of functors from type constructors is used to code up some T-algebras. In *loc.cit.*, there are interesting examples of how to use this kind of inductive types.

Categorical formulation of inductive (coinductive) types can be found in [23, 25]. Related to this is the work [12] where some properties of "algebraic complete" categories are obtained.

The model theory of the inductive types has been a research topic for some time. An old approach is to interpret an inductive type by the initial T-algebra of a κ-continuous functor on SET, the category of sets. This method works also for the generalized inductive types, see [8]. Unfortunately, a monotonic

Fundamenta Informaticae, **26**(2): 115-131, 1996.

†During the preparation of this paper, the author was supported by the CLICS-II project. In the later stage of the preparation, he was under the 863 Project. The author's present address: Department of Computer Science, Shanghai Jiao Tong University, 1954 Hua Shan Road, Shanghai 200030, People's Republic of China.

and κ-bounded functor on the category ω-SET of ω-sets ([18, 29]) in general does not possess an initial T-algebra, the reason being that ω-SET does not have all (filtered) colimits. The ω-SET version of κ-continuous functors is too restrictive to model all generalized inductive types. Another way of interpreting inductive types is to use Aczel's notion of rule sets ([2, 11]). This approach, which is mathematically weaker, does generalize to ω-SET. For more details on both the subjects, see [13, 15].

In this paper, we interpret generalized inductive types in the category PER of partial equivalence relations on ω, the set of natural numbers. What is the advantage of this interpretation over that in ω-SET? First, in the case of PER, the rule sets are more elegantly defined than those in the case of ω-SET. As $\omega \times \omega$ is the largest element in the poset $(Obj(\text{PER}), \subseteq)$, the terminal object $\omega \times \omega$ is somehow the largest 'universe' in the world of per's. On the other hand, we certainly do not have a largest ω-set, whatever that might mean. Second, the existence of such a 'universe' is necessary for the interpretation of coinductive types formulated as terminal T-coalgebras. It is hoped that our model can motivate a general definition of lazy sets in Martin-Löf's type theory. Finally the PER model interprets the *general* inductive types. This is the content of section 6.

2 Generalized Inductive Types

As the basic calculus, we take Martin-Löf's type theory with an infinite hierarchy of cumulative universes. The language, ML^∞, has Π-types and extensional definitional equality (i.e. with the η-rule). Intuitively the small universes satisfy the following properties:

$$Type_0 \in Type_1 \in Type_2 \in \cdots,$$
$$Type_0 \subset Type_1 \subset Type_2 \subset \cdots.$$

In what follows, i ranges over natural numbers. The basic rules are:

$$\frac{}{[]\ \mathbf{valid}}\ \text{Empty Context}$$

$$\frac{\Gamma \vdash A : Type_i}{\Gamma, x : A\ \mathbf{valid}}\ \text{Context}$$

$$\frac{\Gamma, x : A, \Gamma' \vdash \mathbf{valid}}{\Gamma, x : A, \Gamma' \vdash x : A}\ \text{Variable}$$

$$\frac{\Gamma\ \mathbf{valid}}{\Gamma \vdash Type_i : Type_{i+1}}\ \text{Universe}$$

$$\frac{\Gamma \vdash A : Type_i}{\Gamma \vdash A : Type_{i+1}}\ \text{Cumulativity}$$

$$\frac{\Gamma \vdash A : Type_i \quad \Gamma, x : A \vdash B : Type_i}{\Gamma \vdash \Pi x{:}A.B : Type_i}\ \text{Product}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : \Pi x{:}A.B}\ \text{Abstraction}$$

$$\frac{\Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}\ \text{Application}$$

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash A = B : Type_i}{\Gamma \vdash M : B}\ \text{Transitivity}$$

The above description is over simplified. A complete presentation should spell out all the rules about definitional equality and the premises missing in the above formulation. For instance full Application is

$$\frac{\begin{array}{c}\Gamma \vdash A : Type_i \quad \Gamma, x : A \vdash B : Type_i \\ \Gamma \vdash M : \Pi x{:}A.B \quad \Gamma \vdash N : A \quad \Gamma \vdash B[N/x] : Type_i\end{array}}{\Gamma \vdash MN : B[N/x]}.$$

There might appear some redundancy. But it makes much easier the proofs using induction on derivations. We follow the usual convention which is: *1.* to derive $\Gamma \vdash a : A$, one must derive $\Gamma \vdash A : Type_i$ first; *2.* to derive $\Gamma \vdash a = b : A$, one must derive $\Gamma \vdash a : A$ and $\Gamma \vdash b : A$ first; *3.* to derive $\Gamma \vdash A = B : Type_i$, one must derive $\Gamma \vdash A : Type_i$ and $\Gamma \vdash B : Type_i$ first.

One can extend the above calculus in different ways. As our main concern is about the semantics of inductive types, we enrich it with the *generalized inductive types* in [8] to get things off the ground. We will call this language ML^∞_{gi}.

Definition 2.1 *Suppose* $\Gamma, X : Type_i \vdash \phi(X) : Type_i$. *We say* $\phi(X)$ *is* strictly positive *with respect to* X *if* X *does not occur in* ϕ *or* $\phi = X$ *or* $\phi = \Pi x{:}K.\phi'(X)$ *where* X *does not occur in* K *and* $\phi'(X)$ *is strictly positive with respect to* X. *Clearly* $\phi(X)$ *in* $\Gamma, X : Type_i \vdash \phi(X) : Type_i$ *is strictly positive if either* $\phi(X)$ *contains no* X *or* $\phi(X) = \Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.X$ *such that* X *does not occur in any of* $K_1, \cdots, K_m$.

If $\Gamma \vdash A : Type_i$ and $\Gamma \vdash P_A : A \to Type_i$, then P_A can be understood as a property about A. If furthermore $\Gamma, X : Type_i \vdash \phi(X) : Type_i$ and $\phi(X)$ is strictly positive with respect to X and actually contains X, then we can extend P_A to $\Gamma \vdash P_{\phi(A)} : \phi(A) \to Type_i$, a property about $\phi(A)$. This is defined as follows:

- If $\phi(X) = X$, then $P_{\phi(A)} \stackrel{\text{def}}{=} P_A$.
- If $\phi(X) = \Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.X$, then

$$P_{\phi(A)} \stackrel{\text{def}}{=} \lambda z{:}\phi(A).\Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.P_A(zx_1\cdots x_m).$$

Similarly, if $\Gamma \vdash f : \Pi x{:}A.P_A(x)$, we see f as a proof that P_A holds of every inhabitant of A. We extend f to a proof $\phi^\diamond(f)$ showing that $P_{\phi(A)}$ holds of all members of $\phi(A)$,

$$\Gamma \vdash \phi^\diamond(f) : \Pi z{:}\phi(A).P_{\phi(A)}(z) = \Pi z{:}\phi(A).\Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.P_A(zx_1\cdots x_m),$$

in the same fashion:

- If $\phi(X) = X$, then $\phi^\diamond(f) \stackrel{\text{def}}{=} f$.
- If $\phi(X) = \Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.X$, then

$$\phi^\diamond(f) \stackrel{\text{def}}{=} \lambda z{:}\phi(A).\lambda x_1{:}K_1.\cdots\lambda x_m{:}K_m.f(zx_1\cdots x_m).$$

Definition 2.2 *Suppose* $\Gamma, X : Type_i \vdash \Theta(X) : Type_i$. *We say that* Θ *is a* constructor type *if* $\Theta(X)$ *is of the form* $\Pi x_1{:}\phi_1(X).\cdots\Pi x_m{:}\phi_m(X).X$ *with* $\phi_1(X), \cdots, \phi_m(X)$ *being strictly positive with respect to* X.

We are now ready to give rules for generalized inductive types. In all these rules,

$$\Theta_k(X) = \Pi x_{k1}{:}\phi_{k1}(X).\cdots\Pi x_{kn_k}{:}\phi_{kn_k}(X).X$$

is a constructor type for each $k \in \{1, \ldots, n\}$.

Formation

$$\frac{\Gamma, X : Type_i \vdash \Theta_1(X) : Type_i, \ldots, \Gamma, X : Type_i \vdash \Theta_n(X) : Type_i}{\Gamma \vdash \mu X.[\Theta_1(X), \ldots, \Theta_n(X)] : Type_i}$$

We will abbreviate $\mu X.[\Theta_1(X), \ldots, \Theta_n(X)]$ to $\mu X.[\vec{\Theta}]$ or even to μ.

Introduction

$$\frac{\Gamma \vdash t_{k1} : \phi_{k1}(\mu X.[\vec{\Theta}]), \ldots, \Gamma \vdash t_{kn_k} : \phi_{kn_k}(\mu X.[\vec{\Theta}])[t_{k1}, \ldots, t_{kn_k-1}/x_{k1}, \ldots, x_{kn_k-1}]}{\Gamma \vdash \mathsf{intro}_k^\mu(t_{k1}, \ldots, t_{kn_k}) : \mu X.[\Theta_1(X), \ldots, \Theta_n(X)]}$$

for each $k \in \{1, \ldots, n\}$.

Elimination

$$\frac{\begin{array}{l}\Gamma \vdash P_\mu : \mu X.[\Theta_1(X), \ldots, \Theta_n(X)] \to Type_i \\ \Gamma, x_{11} : \phi_{11}(\mu), \ldots, x_{1n_1} : \phi_{1n_1}(\mu), y_{11_1} : P_{\phi_{11_1}(\mu)}(x_{11_1}), \ldots, y_{11_p} : P_{\phi_{11_p}(\mu)}(x_{11_p}) \\ \vdash f_1 : P_\mu(\mathsf{intro}_1^\mu(\vec{x_1})) \\ \vdots \\ \Gamma, x_{n1} : \phi_{n1}(\mu), \ldots, x_{nn_n} : \phi_{nn_n}(\mu), y_{nn_1} : P_{\phi_{nn_1}(\mu)}(x_{nn_1}), \ldots, y_{nn_p} : P_{\phi_{nn_p}(\mu)}(x_{nn_p}) \\ \vdash f_n : P_\mu(\mathsf{intro}_n^\mu(\vec{x_n}))\end{array}}{\Gamma \vdash \mathsf{rec}_\mu(f_1, \ldots, f_n) : \Pi z{:}\mu.P_\mu(z)}$$

This rule calls for some explanation. For each $k \in \{1, \ldots, n\}$, $\phi_{kk_1}(X), \ldots, \phi_{kk_p}(X)$ are among those $\phi_{k1}(X), \ldots, \phi_{kn_k}(X)$ that actually contain the variable X. $f_1, \ldots, f_n$ are induction steps; together they show that the property P_μ holds for every member of $\mu X.[\Theta_1(X), \ldots, \Theta_n(X)]$.

Computation

$$\frac{\begin{array}{l}\Gamma \vdash t_{k1} : \phi_{k1}(\mu), \ldots, \Gamma \vdash t_{kn_k} : \phi_{kn_k}(\mu)[t_{k1}, \ldots, t_{kn_k-1}/x_{k1}, \ldots, x_{kn_k-1}] \\ \Gamma \vdash P_\mu : \mu X.[\Theta_1(X), \ldots, \Theta_n(X)] \rightarrow Type_i \\ \Gamma, x_{11} : \phi_{11}(\mu), \ldots, x_{1n_1} : \phi_{1n_1}(\mu), y_{11_1} : P_{\phi_{11_1}(\mu)}(x_{11_1}), \ldots, y_{11_p} : P_{\phi_{11_p}(\mu)}(x_{11_p}) \\ \vdash f_1 : P_\mu(\mathsf{intro}_1^\mu(\vec{x_1})) \\ \vdots \\ \Gamma, x_{n1} : \phi_{n1}(\mu), \ldots, x_{nn_n} : \phi_{nn_n}(\mu), y_{nn_1} : P_{\phi_{nn_1}(\mu)}(x_{nn_1}), \ldots, y_{nn_p} : P_{\phi_{nn_p}(\mu)}(x_{nn_p}) \\ \vdash f_n : P_\mu(\mathsf{intro}_n^\mu(\vec{x_n}))\end{array}}{\begin{array}{rl}\Gamma \vdash \mathsf{rec}_\mu(\vec{f})(\mathsf{intro}_k^\mu(t_{k1}, \ldots, t_{kn_k})) = & f_k(t_{k1}, \ldots, t_{kn_k}, \\ & (\phi_{kk_1}[t_{k1}, \ldots, t_{kk_1-1}/x_{k1}, \ldots, x_{kk_1-1}])^\diamond(\mathsf{rec}_\mu(\vec{f}))t_{kk_1}, \\ & \cdots, \\ & (\phi_{kk_p}[t_{k1}, \ldots, t_{kk_p-1}/x_{k1}, \ldots, x_{kk_p-1}])^\diamond(\mathsf{rec}_\mu(\vec{f}))t_{kk_p})\end{array}}$$

for each $k \in \{1, \ldots, n\}$. Explicitly, $(\phi_{kk_1}[t_{k1}, \ldots, t_{kk_1-1}/x_{k1}, \ldots, x_{kk_1-1}])^\diamond(\mathsf{rec}_\mu(\vec{f}))t_{kk_1}$ is, say,

$$\begin{array}{l}\lambda x_{kk_1}^1 : K_{kk_1}^1[t_{k1}, \ldots, t_{kk_1-1}/x_{k1}, \ldots, x_{kk_1-1}]. \cdots . \\ \lambda x_{kk_1}^m : K_{kk_1}^m[t_{k1}, \ldots, t_{kk_1-1}/x_{k1}, \ldots, x_{kk_1-1}].\mathsf{rec}_\mu(\vec{f})(t_{kk_1} x_{kk_1}^1 \cdots x_{kk_1}^m).\end{array}$$

Example 2.3 The W-types ([26, 28]). This example is meant to bring out the intuition behind the above rules. There is only one constructor type : $\Theta = \Pi x{:}A.\Pi y{:}B(x) \rightarrow X.X$. Here $\phi_1(X) = A$ and $\phi_2(X) = B(x) \rightarrow X$. The introduction rule is the familiar one:

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B(a) \rightarrow W(A, B)}{\Gamma \vdash \mathsf{sup}(a, b) : W(A, B)}.$$

The reader is advised to check that the elimination rule amounts to

$$\frac{\Gamma \vdash P : W(A, B) \rightarrow Type_i \quad \Gamma, x : A, y : B(x) \rightarrow W(A, B), w : \Pi z{:}B(x).P(yz) \vdash f : P(\mathsf{sup}(x, y))}{\Gamma \vdash \mathsf{rec}_{W(A,B)}(f) : \Pi z{:}W(A, B).P(z)}.$$

The generalized inductive types are straightforward generalizations of W-types. □

3 The Recursive Model

In this section we define a recursive model $\mathcal{M}$ for the language defined in the previous section. The general method follows from [5, 6]. Our contribution is to explore the notion of rule sets to define the extensions of a certain collection of names, which is to be used to interpret the generalized inductive types.

3.1 *Some Preliminaries*

We avail ourselves of an effective pairing function $\langle \_, \_ \rangle$ and two projection functions $()_0$ and $()_1$. The n-tuples will be coded in the usual way. The $(i+1)$-th component of an n-tuple v will be noted by $(v)_i$. We also assume a standard Gödel numbering φ of n-ary recursive functions for each $n \geq 1$. $\varphi_f(a_1, \ldots, a_n)$ will be simplified to $f \cdot (a_1, \ldots, a_n)$.

Definition 3.1 *A* per*, partial equivalence relation, is a transitive symmetric relation A on the set ω of natural numbers. $dom(A) \stackrel{\text{def}}{=} \{n | nAn\}$. $x \in A$ iff xAx. A map from a per A to another per B is the set $[l]_{A \rightarrow B}$ of all natural numbers such that $\forall m, n \in [l]_{A \rightarrow B}.\forall p, q \in A.pAq \Rightarrow (m \cdot p)B(n \cdot q)$, where $x \cdot y$ is the result of applying the x-th recursive function to the number y. We will write $A \preceq B$ if $A \subseteq B$.*

Given pers A and B, we can define the usual constructions as follows. The product $A \times B$ is the per such that $m(A \times B)n$ iff $(m)_0 A (n)_0$ and $(m)_1 B (n)_1$. The exponential $A \rightarrow B$ contains all the pairs $\langle m, n \rangle$ such that for any pair $\langle a, b \rangle \in A$, it is the case that $\langle m \cdot a, n \cdot b \rangle \in B$. The coproduct $A + B$ contains all the pairs $\langle m, n \rangle$ such that either $(m)_0 = (n)_0 = 0$ and $(m)_1 A (n)_1$ or $(m)_0 = (n)_0 = 1$ and $(m)_1 B (n)_1$. The initial object is the empty relation while the terminal object is the total relation. In fact, the category Per is a locally cartesian closed category with finite colimits.

In the sequel we need an 'effective version' of the following definition.

Definition 3.2 *([1])* *A* rule *on the set U is a pair $\frac{u}{v}$ such that $u \subset U$ and $v \in U$. A* rule set *on U is a set of rules on U. Given a rule set R on U, a set A is R-*closed *if for any rule $\frac{u}{v} \in R$, $u \subset A$ implies $v \in A$. The set* inductively defined *by R is the set $\mathcal{I}(R) \stackrel{\text{def}}{=} \bigcap \{A | A \text{ is } R\text{-closed}\}$, the smallest R-closed set. A rule set R is* deterministic *if $\frac{u_1}{v} \in R \wedge \frac{u_2}{v} \in R \Rightarrow u_1 = u_2$.*

Rule sets have been used to interpret inductive types in Martin-Löf's set theory, see [2, 11]. We now briefly explain how to model the generalized inductive types in the category SET. This should serve as a motivation for the PER interpretation to be defined later.

For a type $A(x)$ with a free variable x, we will write $[\![A(x)]\!]_{x:=D}$ for the denotation of $A(x)$ when x is interpreted as D. If a type $A(x_1,\ldots,x_n)$ contains more than one free variables, we will write

$$[\![A(x_1,\ldots,x_n)]\!][D_1,\ldots,D_n/x_1,\ldots,x_n],$$

or even $[\![A(x_1,\ldots,x_n)]\!][D_1,\ldots,D_n]$, for the denotation of $A(x_1,\ldots,x_n)$ when $x_1,\ldots,x_n$ are interpreted as $D_1,\ldots,D_n$ respectively. The notation $[\![A(x,y)]\!][a][y:=b]$, for example, will denote the same thing as $[\![A(x,y)]\!][a,b/x,y]$.

Suppose A is a set and $B : A \longrightarrow \text{SET}$ is a map that associates a set to each member of A. Define $\pi(A,B)$ to be the set of all functions f such that $\forall a{\in}A.f(a){\in}B(a)$. Let $\Theta_j(X) = \Pi x_1{:}\phi_1^j(X).\cdots\Pi x_m{:}\phi_m^j(X).X$ be a constructor type. The rule set R_{Θ_j} is defined as the following set:

$$\left\{ \frac{\cup_{k=1\cdots m} range(f_k)}{\langle n_{\Theta_j}, m, f_1,\ldots,f_m\rangle} \;\middle|\; \begin{array}{l} f_1 \in [\![\phi_1^j(X)]\!][X:=V_\kappa]\wedge\cdots\wedge \\ f_m \in [\![\phi_m^j(X)]\!][f_1,\ldots,f_{m-1}][X:=V_\kappa] \end{array} \right\} \tag{1}$$

where n_{Θ_j} is a code for $\Theta_j(X)$ and V_κ is a large enough universe. For the generalized inductive type $\mu X.[\Theta_1,\ldots,\Theta_n]$, the corresponding rule set $R_{\mu X.[\vec{\Theta}]}$ is the union $\cup_{j=1\cdot\cdot n} R_{\Theta_j}$. The type $\mu X.[\Theta_1,\ldots,\Theta_n]$ can now be interpreted as the smallest $R_{\mu X.[\vec{\Theta}]}$-set $\mathcal{I}(R_{\mu X.[\vec{\Theta}]})$. The eliminator for $\mu X.[\vec{\Theta}]$ can be defined on the way the elements of $\mathcal{I}(R_{\mu X.[\vec{\Theta}]})$ are generated. For details, see *loc.cit.*.

In order to give a recursive model of ML_{gi}^{∞}, we need to explain how the interpretation of generalized inductive types in SET can be carried out in PER.

Suppose A is a per and $B : dom(A) \longrightarrow \text{PER}$ is a map such that for any $m,n \in A$, mAn implies $B(m) = B(n)$. Two new pers are defined as follows:

$$\begin{array}{rcl} \pi(A,B) & \stackrel{\text{def}}{=} & \{\langle f,g\rangle \mid \forall (a,b){\in}A.(f{\cdot}a)B(a)(g{\cdot}b)\}, \\ \sigma(A,B) & \stackrel{\text{def}}{=} & \{\langle\langle a,b\rangle,\langle c,d\rangle\rangle \mid aAc{\wedge}bB(a)d\}. \end{array}$$

In the PER interpretation, a context is interpreted as a per. A type $\Gamma \vdash A$ is modeled by a function $[\![A]\!] : dom([\![\Gamma]\!]) \longrightarrow \text{PER}$ such that $m[\![\Gamma]\!]n$ implies $[\![A]\!](m) = [\![A]\!](n)$. The context $\Gamma, x : A$ is interpreted by $\sigma([\![\Gamma]\!],[\![A]\!])$. $[\![\Gamma \vdash \Pi x{:}A.B]\!]$ is the map $\pi_{[\![\Gamma]\!]}([\![A]\!],[\![B]\!])$ that sends $\alpha \in [\![\Gamma]\!]$ to $\pi([\![A]\!](\alpha,\_),[\![B]\!](\alpha,\_))$. See [17, 19] for more on PER models.

In PER the rule set that corresponds to $\dot{R}_{\Theta_j}$ can be defined as R_{Θ_j}, except that V_κ is replaced by the 'largest' per $\omega{\times}\omega$:

$$\dot{R}_{\Theta_j} \stackrel{\text{def}}{=} \left\{ \frac{\cup_{k=1\cdots m} range(f_k)}{\langle n_{\Theta_j}, m, f_1,\ldots,f_m\rangle} \;\middle|\; \begin{array}{l} f_1 \in [\![\phi_1^j(X)]\!][X:=\omega{\times}\omega]\wedge\cdots\wedge \\ f_m \in [\![\phi_m^j(X)]\!][f_1,\ldots,f_{m-1}][X:=\omega{\times}\omega] \end{array} \right\}. \tag{2}$$

In this definition, $[\![\phi_k^j(X)]\!][f_1,\ldots,f_{k-1}][X:=\omega{\times}\omega]$ is just $[\![\phi_k^j(X)]\!][f_1,\ldots,f_{k-1}]$ if X does not occur in $\phi_k^j(X)$; and in this case $range(f_k)$ is the empty set. On the other hand if $\phi_k^j(X)$ does contain X, say $\phi_k^j(X) = \Pi x_{k1}{:}K_{k1}^j.\cdots\Pi x_{km_k}{:}K_{km_k}^j.X$, then

$$\begin{array}{rl} & [\![\phi_k^j(X)]\!][f_1,\ldots,f_{k-1}][X:=\omega{\times}\omega] \\ = & \pi([\![K_{k1}^j]\!][f_1,\ldots,f_{k-1}],\ldots,\pi([\![K_{km_k}^j]\!][f_1,\ldots,f_{k-1}],\omega{\times}\omega)\cdots). \end{array}$$

Let $\dot{R}_{\mu X.[\vec{\Theta}]} \stackrel{\text{def}}{=} \cup_{i=1\cdot\cdot n}\dot{R}_{\Theta_i}$. $\dot{R}_{\mu X.[\vec{\Theta}]}$ is a rule set on ω. So $\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})$ is a subset of ω. But we want a per. This per $\mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}))$ can be obtained as the least fixpoint of the following sequence:

- $\mathcal{P}^0$ is defined to be the initial per $\emptyset$.
- $\mathcal{P}^{\alpha+1} \stackrel{\text{def}}{=} \cup_{j=1\cdot\cdot n} S_j$ where S_j is the following set:

$$\left\{ \begin{array}{l} \langle\langle n_{\Theta_j}, m, f_1,\ldots,f_m\rangle, \\ \langle n_{\Theta_j}, m, g_1,\ldots,g_m\rangle\rangle \end{array} \;\middle|\; \begin{array}{l} \langle n_{\Theta_j}, m, f_1,\ldots,f_m\rangle \in \mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})\wedge \\ \langle n_{\Theta_j}, m, g_1,\ldots,g_m\rangle \in \mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})\wedge \\ f_1\{[\![\phi_1^j(X)]\!][X:=\mathcal{P}^\alpha]\}g_1\wedge\cdots\wedge \\ f_m\{[\![\phi_m^j(X)]\!][f_1,\ldots,f_{m-1}][X:=\mathcal{P}^\alpha]\}g_m \end{array} \right\}.$$

  Clearly, $\mathcal{P}^\alpha \subset \mathcal{P}^{\alpha+1} \subset \mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}){\times}\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})$.

- For a limit ordinal θ, $\mathcal{P}^\theta \stackrel{\text{def}}{=} \cup_{\alpha\in\theta}\mathcal{P}^\alpha$.

We can now interpret $\mu X.[\vec{\Theta}]$ by $\mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}))$. Suppose we have the following derivation using the introduction rule:

$$\frac{\vdash t_1 : \phi_1^j(\mu X.[\vec{\Theta}]),\ \cdots\ \vdash t_m : \phi_m^j[t_1,\ldots,t_{m-1}/x_1,\ldots,x_{m-1}](\mu X.[\vec{\Theta}])}{\vdash \mathsf{intro}_j^\mu(t_1,\ldots,t_m) : \mu X.[\Theta_1,\ldots,\Theta_n]}.$$

By induction hypothesis, we have

$$\begin{aligned}
[\![t_1]\!] &\in [\![\phi_1^j(X)]\!][X := \mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}))],\\
&\vdots\\
[\![t_m]\!] &\in [\![\phi_m^j(X)]\!][[\![t_1]\!],\ldots,[\![t_{m-1}]\!]][X := \mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}))].
\end{aligned}$$

Then by the definition of $\mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}))$, $\langle n_{\Theta_j}, m, [\![t_1]\!],\ldots,[\![t_m]\!]\rangle \in \mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}))$.

3.2 *The Model*

The purpose of this section is to construct a model $\mathcal{M}$ on the partial applicative structure $(\omega,\cdot)$. In section 4 an interpretation of ML_{gi}^∞ in $\mathcal{M}$ will be given. $\mathcal{M}$ consists of a countable set of *names*, each of which is associated with a per on the set ω of natural numbers, called its *extension*. As our language $\mathcal{L}$ has an infinite hierarchy of small universes, $\mathcal{M}$ must contain an infinite cumulative recursive submodels $\mathcal{M}_0,\mathcal{M}_1,\cdots,\mathcal{M}_i,\cdots$. *In each of the following definitions, one first postulates a name, one then attaches a subset of ω to the name, finally one defines an equivalence relation on the subset.* There are three kinds of judgements:

- Suppose $\mathcal{M}_0,\ldots,\mathcal{M}_{i-1}$ have already been defined. $\mathcal{M}_i \models \langle l,m\rangle$ means that $\langle l,m\rangle$ is a name already defined in $\mathcal{M}_i$. The number l classifies the names of a particular collection of per's.
- $\mathcal{M}_i \models ext(n) = S$ says that n names S, a subset of ω.
- $\mathcal{M}_i \models x = y \in ext(n)$ means that x and y are equivalent in $\mathcal{M}_i$.

$\mathcal{M}$ is inductively defined as follows:

1. Let $\mathcal{M}_0 \models \langle 0,1\rangle$, $\mathcal{M}_0 \models ext(\langle 0,1\rangle) = \omega$, and $\mathcal{M}_0 \models x = y \in ext(\langle 0,1\rangle)$ for all $x,y\in\omega$. So $\langle 0,1\rangle$ names the terminal per.
2. If $\mathcal{M}_0 \models a$ and $\forall x(\mathcal{M}_0 \models x\in ext(a) \Rightarrow \mathcal{M}_0 \models d{\cdot}x)$ and $\forall x,y(\mathcal{M}_0 \models x = y \in ext(a) \Rightarrow \mathcal{M}_0 \models d{\cdot}x \doteq d{\cdot}y)$, then d is said to be a *family of names* in $\mathcal{M}_0$ *over* a. For the definition of $\doteq$, see 5 below.
3. Suppose b is a family of names in $\mathcal{M}_0$ over a. Let $\pi(a,b) \stackrel{\text{def}}{=} \langle 1,a,b\rangle$. Then define $\mathcal{M}_0 \models \pi(a,b)$, and $\mathcal{M}_0 \models n\in ext(\pi(a,b))$ iff

   $$\begin{aligned}&\forall x(\mathcal{M}_0 \models x\in ext(a) \Rightarrow \mathcal{M}_0 \models n{\cdot}x \in ext(b{\cdot}x))\wedge\\ &\forall x,y(\mathcal{M}_0 \models x = y\in ext(a) \Rightarrow \mathcal{M}_0 \models n{\cdot}x = n{\cdot}y \in ext(b{\cdot}x))\end{aligned}$$

   and $\mathcal{M}_0 \models m = n\in ext(\pi(a,b))$ iff

   $$\begin{aligned}&\mathcal{M}_0 \models m\in ext(\pi(a,b))\wedge\mathcal{M}_0 \models n\in ext(\pi(a,b))\wedge\\ &\forall x(\mathcal{M}_0 \models x\in ext(a) \Rightarrow \mathcal{M}_0 \models m{\cdot}x = n{\cdot}x \in ext(b{\cdot}x)).\end{aligned}$$

4. Let's abbreviate $\langle 2,n,\langle n_1,l_1,c^1\rangle,\cdots,\langle n_n,l_n,c^n\rangle\rangle$ to ind, where $c^1,c^2,\cdots,c^n$ are names already defined. Define

   $$\mathcal{M}_0 \models ind, \quad \mathcal{M}_0 \models ext(ind) = \mathcal{P}(\mathcal{I}(R_{ind}))$$

   where $R_{ind} = \cup_{k=1\cdot\cdot n} R_{\langle n_k,l_k,c^k\rangle}$. $R_{\langle m,l,d\rangle}$ is defined as follows (if $m = n_k$, then $intro$ is $intro_k$, a code for the elements of $ext(ind)$ of a particular form):

   $$R_{\langle m,l,d\rangle} \stackrel{\text{def}}{=} \left\{ \frac{\cup_{i\in\{1,\ldots,m\}} range(t_i,(l)_{i-1},(d^i)_1)}{\langle intro,m,t_1,\cdots,t_m\rangle} \;\middle|\; \begin{array}{l}(d^1)_0 = 1\wedge\mathcal{M}_0 \models t_1\in ext((d^1)_1)\\ \wedge(d^2)_0 = 1\wedge\mathcal{M}_0 \models t_2\in ext((d^2)_1)\\ \wedge\cdots\wedge(d^m)_0 = 1\wedge\\ \mathcal{M}_0 \models t_m\in ext((d^m)_1)\end{array} \right\}$$

where $d^1 \stackrel{\text{def}}{=} d, d^2 \stackrel{\text{def}}{=} (d^1)_2{\cdot}t_1, \cdots, d^m \stackrel{\text{def}}{=} (\cdots((d^1)_2{\cdot}t_1)_2{\cdot}t_2\cdots)_2{\cdot}t_{m-1}$ and

$$range(t,0,d) \stackrel{\text{def}}{=} \emptyset,$$

$$range(t,l+1,d) \stackrel{\text{def}}{=} \left\{ t{\cdot}a_1{\cdot}\cdots{\cdot}a_l \;\middle|\; \begin{array}{l} (d)_0 = 1 \wedge \mathcal{M}_0 \models a_1 \in ext((d)_1) \wedge \\ ((d)_2{\cdot}a_1)_0 = 1 \wedge \mathcal{M}_0 \models a_2 \in ext(((d)_2{\cdot}a_1)_1) \\ \wedge\cdots\wedge((\cdots((d)_2{\cdot}a_1)_2{\cdot}a_2\cdots)_2{\cdot}a_{n-1})_0 = 1 \\ \wedge \mathcal{M}_0 \models a_l \in ext(((\cdots((d)_2{\cdot}a_1)_2{\cdot}a_2\cdots)_2{\cdot}a_{l-1})_1) \\ \wedge((\cdots((d)_2{\cdot}a_1)_2{\cdot}a_2\cdots)_2{\cdot}a_{l-1})_2{\cdot}a_l = \langle 0,1\rangle \end{array} \right\}$$

provided that $\mathcal{M}_0 \models t \in ext(d)$. Notice that when $l = 0$, $t{\cdot}a_1\cdots a_l$ is just t. As the set of conclusions of $\cup_{k=1\cdots n} range(n_k, c^k)$ is closed, every element of $ext(ind)$ is of the form $\langle intro_k, n_k, t_1, \cdots, t_{n_k}\rangle$ for some $k \in \{1,\ldots,n\}$.

5. $\mathcal{M}_0 \models a \doteq b$ if $\mathcal{M}_0 \models a$ and $\mathcal{M}_0 \models b$ and $\forall x(\mathcal{M}_0 \models x \in ext(a) \Leftrightarrow \mathcal{M}_0 \models x \in ext(b))$ and $\forall x,y(\mathcal{M}_0 \models x = y \in ext(a) \Leftrightarrow \mathcal{M}_0 \models x = y \in ext(b))$. So $\doteq$ is the extensional equality. This completes our definition of $\mathcal{M}_0$.

6. $\mathcal{M}_1 \models \langle 3,0\rangle$, the name of the first universe. $(\mathcal{M}_1 \models x \in ext(\langle 3,0\rangle) \Leftrightarrow \mathcal{M}_0 \models x)$ and $(\mathcal{M}_1 \models x = y \in ext(\langle 3,0\rangle) \Leftrightarrow \mathcal{M}_0 \models x \doteq y)$.

7. Let $\mathcal{M}_1$ include $\mathcal{M}_0$. Therefore we have intuitively $\mathcal{M}_0 \subset \mathcal{M}_1$ and $\mathcal{M}_0 \in \mathcal{M}_1$.

8. We then close up $\mathcal{M}_1$ the same way we did to $\mathcal{M}_0$.

9. Similarly, we can construct $\mathcal{M}_2, \mathcal{M}_3, \cdots$. For instance let $\mathcal{M}_2 \models \langle 3,1\rangle, \mathcal{M}_3 \models \langle 3,2\rangle, \cdots$ etc. This completes the definition of $\mathcal{M}_i$ for $i \in \omega$.

10. Let b be a family of names in $\mathcal{M}_i$ over a. Then $\mathcal{M} \models \gamma(a,b)$ where $\gamma(a,b) \stackrel{\text{def}}{=} \langle 4,a,b\rangle$.

$$\mathcal{M} \models (c^0,c^1) \in ext(\gamma(a,b)) \Leftrightarrow \mathcal{M} \models c^0 \in ext(a) \wedge \mathcal{M} \models c^1 \in ext(b{\cdot}c^0),$$

$$\mathcal{M} \models (c^0,c^1) = (d^0,d^1) \in ext(\gamma(a,b)) \Leftrightarrow \mathcal{M} \models c^0 = d^0 \in ext(a) \wedge \mathcal{M} \models c^1 = d^1 \in ext(b{\cdot}c^0).$$

We say the length of $\gamma(a,b)$, $len(\gamma(a,b))$, is 2.

11. Let $\gamma' = \gamma(\cdots\gamma(\gamma(a_1,a_2),a_3),\ldots,a_n)$. Suppose $\mathcal{M} \models \gamma'$. A *family* f of names over γ', noted $\mathcal{M} \models f : \gamma' \longrightarrow \langle 3,i\rangle$, is an index of an n-ary recursive function such that if $\mathcal{M} \models (c^0,\ldots,c^{n-1}) \in ext(\gamma')$ then $\mathcal{M}_{i+1} \models f{\cdot}(c^0,\ldots,c^{n-1}) \in ext(\langle 3,i\rangle)$ and if $\mathcal{M} \models (c^0,\ldots,c^{n-1}) = (d^0,\ldots,d^{n-1}) \in ext(\gamma')$, then $\mathcal{M}_{i+1} \models f{\cdot}(c^0,\ldots,c^{n-1}) = f{\cdot}(d^0,\ldots,d^{n-1}) \in ext(\langle 3,i\rangle)$.

12. Suppose $len(\gamma(a,b)) = n$. $\mathcal{M} \models f = g : \gamma(a,b) \longrightarrow \langle 3,i\rangle$ iff $\mathcal{M} \models f : \gamma(a,b) \longrightarrow \langle 3,i\rangle$ and $\mathcal{M} \models g : \gamma(a,b) \longrightarrow \langle 3,i\rangle$ and $\mathcal{M} \models (c^0,\ldots,c^{n-1}) \in ext(\gamma(a,b)) \Rightarrow \mathcal{M}_{i+1} \models f{\cdot}(c^0,\ldots,c^{n-1}) = g{\cdot}(c^0,\ldots,c^{n-1}) \in ext(\langle 3,i\rangle)$.

13. Suppose $\mathcal{M} \models f : \gamma(a,b) \longrightarrow \langle 3,i\rangle$. Then $\mathcal{M} \models \gamma(\gamma(a,b),f)$ and $len(\gamma(\gamma(a,b),f))$ is $len(\gamma(a,b)) + 1$.

$$\begin{array}{ll} & \mathcal{M} \models (c^0,\ldots,c^{n-1},c^n) \in ext(\gamma(\gamma(a,b),f)) \\ \Leftrightarrow & \mathcal{M} \models (c^0,\ldots,c^{n-1}) \in ext(\gamma(a,b)) \wedge \mathcal{M}_i \models c^n \in ext(f{\cdot}(c^0,\ldots,c^{n-1})), \\ & \mathcal{M} \models (c^0,\ldots,c^{n-1},c^n) = (d^0,\ldots,d^{n-1},d^n) \in ext(\gamma(\gamma(a,b),f)) \\ \Leftrightarrow & \mathcal{M} \models (c^0,\ldots,c^{n-1}) = (d^0,\ldots,d^{n-1}) \in ext(\gamma(a,b)) \\ & \wedge \mathcal{M}_i \models c^n = d^n \in ext(f{\cdot}(c^0,\ldots,c^{n-1})). \end{array}$$

14. Suppose $\mathcal{M} \models A : \gamma \longrightarrow \langle 3,i\rangle$. Then $\mathcal{M} \models a : \gamma \Longrightarrow A$ iff for any $x \in ext(\gamma)$, $a{\cdot}x \in ext(A{\cdot}x)$. And $\mathcal{M} \models a = b : \gamma \Longrightarrow A$ iff for any $x \in ext(\gamma)$, $a{\cdot}x = b{\cdot}x \in ext(A{\cdot}x)$.

This completes our definition of $\mathcal{M}$.

4 The Interpretation

The interpretation $\hat{}: \mathcal{L} \longrightarrow \mathcal{M}$ is defined inductively on the derivations of terms. A valid context is interpreted by a name of the form $\gamma(\_,\_)$. A sequent $x_1 : A_1, \ldots, x_n : A_n \vdash a : A$ (or equivalently just a) is interpreted as an (equivalence class of) n-ary recursive function. A type (under a context) is modeled by either a name or the extension of a name depending on whether the type appears on the left hand side or right hand side of ':'. We will write $\hat{A}$ for $\hat{A}$ when A is a long expression. We also use Kleene's notation $\Lambda x.t$ for an index of the function $\lambda x.t$.

- Empty Context. The empty context is interpreted by $\langle 0, 1\rangle$.
- Context. By induction hypothesis, $\mathcal{M} \models \hat{A} : \hat{\Gamma} \longrightarrow \langle 3, i\rangle$. $\widehat{\Gamma, x : A} \stackrel{\text{def}}{=} \gamma(\hat{\Gamma}, \hat{A})$.
- Variable. $x_1 : A_1, \ldots, x_i : A_i, \ldots, x_n : A_n \vdash x_i : A_i$ is interpreted by the recursive function $\Lambda x_1, \ldots, x_n.x_i$.
- Universe. $\widehat{Type_i} \stackrel{\text{def}}{=} \Lambda x_0.\cdots\Lambda x_{n-1}.\langle 3, i\rangle$. Clearly $\Lambda x_0.\cdots\Lambda x_{n-1}.\langle 3, i\rangle$ is a family of names.
- Cumulativity. This rule is valid because of the cumulativity of $\mathcal{M}_0,\ \mathcal{M}_1, \ldots$.
- Product. By induction hypotheses, $\mathcal{M} \models \hat{A} : \hat{\Gamma} \longrightarrow \langle 3, i\rangle$, and $\mathcal{M} \models \hat{B} : \gamma(\hat{\Gamma}, \hat{A}) \longrightarrow \langle 3, i\rangle$. Suppose $\Gamma = x_1 : A_1, \ldots, x_i : A_i, \ldots, x_n : A_n$. By s-m-n theorem, there is a total recursive function h such that
$$\hat{B}\cdot(x_1, \ldots, x_n, x) = h(\hat{B}, x_1, \ldots, x_n)\cdot x.$$
For $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$, $h(\hat{B}, a^0, \ldots, a^{n-1})$ is a family of names over $\hat{\Gamma}$ for if $\mathcal{M}_i \models a \in ext(\hat{A}\cdot(a^0, \ldots, a^{n-1}))$, $\mathcal{M}_i \models h(\hat{B}, a^0, \ldots, a^{n-1})\cdot a = \hat{B}\cdot(a^0, \ldots, a^{n-1}, a)$. The procedure that sends $a^0, \ldots, a^{n-1}$ to $\pi(\hat{A}(a^0, \ldots, a^{n-1}), h(\hat{B}, a^0, \ldots, a^{n-1}))$ is obviously an n-ary recursive function. Assign to $(\Gamma \vdash \Pi x{:}A.B : Type_i)\hat{}$ an index for that function.
- Abstraction. By induction hypothesis, $\mathcal{M} \models \hat{M} : \gamma(\hat{\Gamma}, \hat{A}) \Longrightarrow \hat{B}$. Suppose $len(\hat{\Gamma}) = n$. Again by s-m-n theorem, there is a total recursive function g such that
$$\hat{M}\cdot(x_1, \ldots, x_n, x) = g(\hat{M}, x_1, \ldots, x_n)\cdot x.$$
Let $\lambda\widehat{x{:}A.M}$ be this g. We need to show that $\mathcal{M} \models \lambda\widehat{x{:}A.M} : \hat{\Gamma} \Longrightarrow \Pi\widehat{x{:}A.B}$. By induction hypothesis, $\mathcal{M} \models \hat{A} : \hat{\Gamma} \longrightarrow \langle 3, i\rangle$, $\mathcal{M} \models \hat{B} : \gamma(\hat{\Gamma}, \hat{A}) \longrightarrow \langle 3, i\rangle$, $\mathcal{M} \models \hat{M} : \gamma(\hat{\Gamma}, \hat{A}) \Longrightarrow \hat{B}$. Suppose $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$. For any a such that $\mathcal{M} \models a \in ext(\hat{A}\cdot(a^0, \ldots, a^{n-1}))$, we have $\lambda\widehat{x{:}A.M}\cdot a = g(\hat{M}, a^0, \ldots, a^{n-1})\cdot a = \hat{M}\cdot(a^0, \ldots, a^{n-1}, a)$. But $\mathcal{M}_i \models \hat{M}\cdot(a^0, \ldots, a^{n-1}, a) \in ext(\hat{B}\cdot(a^0, \ldots, a^{n-1}, a))$. It follows that $\mathcal{M}_i \models \lambda\widehat{x{:}A.M}\cdot a \in ext(h(\hat{B}, a^0, \ldots, a^{n-1})\cdot a)$ where h is obtained in the previous case. The second condition in the definition of γ can be similarly verified. Therefore $\mathcal{M} \models \lambda\widehat{x{:}A.M} : \hat{\Gamma} \Longrightarrow \Pi\widehat{x{:}A.B}$.
- Application. By induction hypotheses, $\mathcal{M} \models \hat{M} : \hat{\Gamma} \Longrightarrow \Pi\widehat{x{:}A.B}$ and $\mathcal{M} \models \hat{N} : \hat{\Gamma} \Longrightarrow \hat{A}$. $\widehat{M\cdot N}$ is interpreted as $\Lambda x_1, \ldots, x_n.(\hat{M}\cdot(x_1, \ldots, x_n))\cdot(\hat{N}\cdot(x_1, \ldots, x_n))$.
- Transitivity. This rule is obviously valid in the model.
- Formation. By induction hypothesis, $\mathcal{M} \models \widehat{\Theta_k} : \gamma(\hat{\Gamma}, \Lambda x_0, \ldots, x_{n-1}.\langle 3, i\rangle) \longrightarrow \langle 3, i\rangle$. Because $\mathcal{M}_{i+1} \models \langle 0, 1\rangle \in ext(\langle 3, i\rangle)$, $\mathcal{M}_{i+1} \models \widehat{\Theta_k}\cdot(a^0, \ldots, a^{n-1}, \langle 0, 1\rangle) \in ext(\langle 3, i\rangle)$ for $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$. We assign to $\hat{\mu}_{(a^0,\ldots,a^{n-1})}$ the number
$$\langle 2, n, \langle n_1, l_1, \widehat{\Theta_1}\cdot(a^0, \ldots, a^{n-1}, \langle 0, 1\rangle)\rangle, \ldots, \langle n_n, l_n, \widehat{\Theta_n}\cdot(a^0, \ldots, a^{n-1}, \langle 0, 1\rangle)\rangle\rangle,$$
where $l_i = \langle l_i^1, \ldots, l_i^{n_i}\rangle$ for $i \in \{1, \ldots, n\}$. Here for $o \in \{1, \ldots, n_i\}$, l_i^o is $q + 1$ if $\phi_{io}(X)$ is the type $\Pi z_1{:}K_1 \cdots \Pi z_q{:}K_q.X$ and is 0 if $\phi_{io}(X)$ does not contain X. So $\hat{\mu}$ is $\Lambda x_1, \ldots, x_n.\hat{\mu}_{(x_1,\ldots,x_n)}$.
- Introduction. We have already explained how the terms of introduction form should be interpreted.

- Elimination and Computation. By induction hypothesis, for any $k \in \{1,\ldots,n\}$ and any $\mathcal{M} \models (a^0,\ldots,a^{n-1}) \in ext(\hat{\Gamma})$,

$$\begin{array}{rcl} g_k(x,y) & \stackrel{\text{def}}{=} & \hat{f}_k(a^0,\ldots,a^{n-1},(x)_2,\ldots,(x)_{n_k+1},\widehat{\phi^{\diamond}_{kk_1}}[(x)_2,\ldots,(x)_{k_1}]\cdot(y)\cdot(x)_{k_1+1}, \\ & & \ldots,\widehat{\phi^{\diamond}_{kk_p}}[(x)_2,\ldots,(x)_{k_p}]\cdot(y)\cdot(x)_{k_p+1}) \end{array}$$

is a recursive function. By Church's thesis, the following function

$$f_{(a^0,\ldots,a^{n-1})}(x,y) \stackrel{\text{def}}{=} \left\{ \begin{array}{lll} g_1(x,y); & \textit{if} & (x)_0 = intro_1 \wedge (x)_1 = n_1 \\ g_2(x,y); & \textit{if} & (x)_0 = intro_2 \wedge (x)_1 = n_2 \\ & \vdots & \\ g_n(x,y); & \textit{if} & (x)_0 = intro_n \wedge (x)_1 = n_n \end{array} \right.$$

is a recursive function. Besides there is a Turing machine that takes g_1-th, g_2-th,... and g_n-th Turing machines and constructs the $f_{(a^0,\ldots,a^{n-1})}$-th Turing machine. To see that, one notices that all such a machine has to do is to plug the g_1-th, g_2-th,... and g_n-th Turing machines into appropriate places in a machine that computes a branching programme; then it imitates a universal Turing machine and starts generating $(n+2)$-ary Turing machine; upon generating the i-th machine, it compares it to the composed one token by token; when it recognizes the m-th machine, it returns m and stops. By recursion theorem, there is (an index of) a recursive function $r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})})$ such that $r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})})\cdot x \simeq f_{(a^0,\ldots,a^{n-1})}(x, r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})}))$ where $f \simeq g$ iff $(f\uparrow \wedge g\uparrow)$ or $(f\downarrow \Leftrightarrow g\downarrow)\wedge(f\downarrow \Rightarrow f = g)$. Interpret $\mathsf{rec}_\mu(f_1,\ldots,f_n)$ by $\Lambda x_1,\ldots,x_n.r_{(x_1,\ldots,x_n)}(f_{(x_1,\ldots,x_n)})$. Now we have to argue that

$$\mathcal{M} \models \Lambda x_1,\ldots,x_n.r_{(x_1,\ldots,x_n)}(f_{(x_1,\ldots,x_n)}) : \hat{\Gamma} \Longrightarrow \Pi z{:}\widehat{\mu.P_\mu}(z).$$

Given $\mathcal{M} \models (a^0,\ldots,a^{n-1}) \in ext(\hat{\Gamma})$,

$$\begin{array}{rcl} & & (\Lambda x_1,\ldots,x_n.r_{(x_1,\ldots,x_n)}(f_{(x_1,\ldots,x_n)}))\cdot(a^0,\ldots,a^{n-1}) \\ & = & r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})}). \end{array}$$

For any $\mathcal{M} \models t \in ext(\hat{\mu}_{(a^0,\ldots,a^{n-1})})$, t must be of the form

$$\langle intro_k, n_k, t_{k1},\ldots,t_{kn_k}\rangle$$

for some $k \in \{1,\ldots,n\}$. Then

$$\begin{array}{rcl} & & r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})})t \\ & \simeq & f_{(a^0,\ldots,a^{n-1})}(t, r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})})) \\ & = & \hat{f}_k(a^0,\ldots,a^{n-1},t_{k1},\ldots,t_{kn_k},\widehat{\phi^{\diamond}_{kk_1}}[t_{k1},\ldots,t_{kk_1-1}](r_{(a^0,\ldots,a^{n-1})} \\ & & (f_{(a^0,\ldots,a^{n-1})}))t_{k_1},\ldots, \\ & & \widehat{\phi^{\diamond}_{kk_p}}[t_{k1},\ldots,t_{kk_p-1}](r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})}))t_{k_p}) \end{array}$$

where by definition $\widehat{\phi^{\diamond}_{kk_1}}[t_{k1},\ldots,t_{kk_1-1}](r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})}))t_{k_1}$ is, say

$$\Lambda x_1,\ldots,x_m.r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})})(t_{kk_1}x_1\cdots x_m).$$

We are reduced to show that $r_{(a^0,\ldots,a^{n-1})}(f_{(a^0,\ldots,a^{n-1})})(t_{kk_1}x_1\cdots x_m)$ etc. are the right kind of functions. But this is the induction principle associated with the definition of the corresponding rule set.

- Finally, let's explain how substitution is interpreted. Suppose $\Gamma \vdash N : A$ and $\Gamma, x : A \vdash M : B$. Then $\Gamma \vdash M[N/x] : B[N/x]$ is interpreted by $\mathcal{M} \models \hat{M}\circ\langle id,\hat{N}\rangle : \hat{\Gamma} \Rightarrow \widehat{B[N/x]}$ where $\hat{M}\circ\langle id,\hat{N}\rangle$ is the n-ary recursive function such that if $\mathcal{M} \models (a^0,\ldots,a^{n-1}) \in ext(\hat{\Gamma})$, then $(\hat{M}\circ\langle id,\hat{N}\rangle)\cdot(a^0,\ldots,a^{n-1}) = \hat{M}\cdot(a^0,\cdots,a^{n-1},\hat{N}\cdot(a^0,\ldots,a^{n-1}))$. Similarly, $\Gamma \vdash B[N/x] : Type_i$ is interpreted by $\mathcal{M} \models \hat{B}\circ\langle id,\hat{N}\rangle : \hat{\Gamma} \Longrightarrow \langle 3,i\rangle$ such that $(\hat{B}\circ\langle id,\hat{N}\rangle)\cdot(a^0,\ldots,a^{n-1}) = \hat{B}\cdot(a^0,\ldots,a^{n-1},\hat{N}\cdot(a^0,\ldots,a^{n-1}))$. Notice that $\mathcal{M} \models \hat{M}\cdot(a^0,\ldots,a^{n-1},\hat{N}\cdot(a^0,\ldots,a^{n-1})) \in ext(\hat{B}\cdot(a^0,\ldots,a^{n-1},\hat{N}\cdot(a^0,\ldots,a^{n-1})))$ follows from $\mathcal{M} \models \hat{M} : \gamma(\hat{\Gamma},\hat{A}) \Longrightarrow \hat{B}$.

This completes the definition of the interpretation. Let us consider the β-rule:

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash N : A}{\Gamma \vdash (\lambda x{:}A.M){\cdot}N = M[n/x] : B[N/x]}.$$

By definition $\widehat{\lambda x{:}A.M} \stackrel{\text{def}}{=} g$ where g is obtained from s-m-n theorem such that if $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$, then $(g{\cdot}(a^0, \ldots, a^{n-1})){\cdot}(\hat{N}{\cdot}(a^0, \ldots, a^{n-1})) = \hat{M}{\cdot}(a^0, \ldots, a^{n-1}, \hat{N}{\cdot}(a^0, \ldots, a^{n-1}))$. But $\hat{M}{\cdot}(a^0, \ldots, a^{n-1}, \hat{N}{\cdot}(a^0, \ldots, a^{n-1})) = \widehat{M[N/x]}{\cdot}(a^0, \ldots, a^{n-1})$ and $(g{\cdot}(a^0, \ldots, a^{n-1})){\cdot}(\hat{N}{\cdot}(a^0, \ldots, a^{n-1})) = \widehat{((\lambda x{:}A.M){\cdot}N)}{\cdot}(a^0, \ldots, a^{n-1})$.

Now we explain why the interpretation is sound with respect to the η-rule:

$$\frac{\Gamma \vdash f : \Pi x{:}A.B}{\Gamma \vdash f = \lambda x{:}A.fx : \Pi x{:}A.B}.$$

By induction hypothesis, $\mathcal{M} \models \hat{f} : \hat{\Gamma} \Longrightarrow \widehat{\Pi x{:}A.B}$. From $\hat{f}$ one can effectively obtain an (n+1)-ary recursive function $\hat{f}'$ such that for $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$, $\mathcal{M}_i \models a^n \in ext(\hat{A}{\cdot}(a^0, \ldots, a^{n-1}))$, $\hat{f}'{\cdot}(a^0, \ldots, a^{n-1}, a^n) = \hat{f}{\cdot}(a^0, \ldots, a^{n-1})$. Here $\hat{f}'$ is the denotation of $\Gamma, x : A \vdash f : \Pi x{:}A.B$. We also have $\mathcal{M} \models \hat{x} : \gamma(\hat{\Gamma}, \hat{A}) \Longrightarrow \hat{A}$. By definition, $\lambda x{:}A.fx$ is interpreted by an n-ary recursive function g such that

$$(\hat{f}'{\cdot}(a^0, \ldots, a^{n-1}, a^n)){\cdot}(\hat{x}{\cdot}(a^0, \ldots, a^{n-1}, a^n)) = (g{\cdot}(a^0, \ldots, a^{n-1})){\cdot}a^n.$$

But the left hand side of the above equation is equal to $(\hat{f}{\cdot}(a^0, \ldots, a^{n-1})){\cdot}a^n$ by definition. It follows that

$$\mathcal{M}_i \models g{\cdot}(a^0, \ldots, a^{n-1}) = \hat{f}{\cdot}(a^0, \ldots, a^{n-1}) \in ext((\widehat{\Pi x{:}A.B}){\cdot}(a^0, \ldots, a^{n-1})).$$

Therefore $\mathcal{M} \models g = \hat{f} : \hat{\Gamma} \Longrightarrow \widehat{\Pi x{:}A.B}$.

5 Positivity Need Not be Strict

If $X : Type_i \vdash \phi(X) : Type_i$ and $\phi(X)$ is positive (instead of strictly positive), the definition of P_A and $f^\diamond$ remains virtually unchanged.

- If $\phi(X) = X$, then $P_{\phi(A)} \stackrel{\text{def}}{=} P_A$ and $\phi^\diamond(f) \stackrel{\text{def}}{=} f$.
- If $\phi(X) = \Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.X$, then

$$\begin{aligned} P_{\phi(A)} &\stackrel{\text{def}}{=} \lambda z{:}\phi(A).\Pi x_1{:}K_1[A/X].\cdots\Pi x_m{:}K_m[A/X].P_A(zx_1\cdots x_m), \\ \phi^\diamond(f) &\stackrel{\text{def}}{=} \lambda z{:}\phi(A).\lambda x_1{:}K_1[A/X].\cdots\lambda x_m{:}K_m[A/X].f(zx_1\cdots x_m). \end{aligned}$$

One is tempted to relax the strict positivity condition in the definition of generalized inductive types. This is reinforced by the fact that what we have given is a sound interpretation of this more liberal inductive types. Let's first see an example. Suppose $\Theta_1(X) = \Pi x{:}\phi(X).X$, $\Theta_2(X) = X$ and $\phi(X) = \Pi y{:}X \to A.X$. Some of the rules concerned with $\mu X.[\Theta_1, \Theta_2]$ are

$$\frac{}{\vdash \bot : \mu} \qquad \frac{\vdash t : (\mu \to A) \to \mu}{\vdash intro(t) : \mu}$$

$$\frac{\begin{array}{c}\vdash P : \mu \to Type_i \quad \vdash f_0 : P(\bot) \\ x : (\mu \to A) \to \mu, y : \Pi z{:}\mu \to A.P(xz) \vdash f_1 : P(intro(x))\end{array}}{\vdash \mathsf{rec}(f_0, f_1)(\bot) = f_0 : P(\bot)}$$

$$\frac{\begin{array}{c}\vdash P : \mu \to Type_i \quad \vdash t : (\mu \to A) \to \mu \quad \vdash f_0 : P(\bot) \\ x : (\mu \to A) \to \mu, y : \Pi z{:}\mu \to A.P(xz) \vdash f_1 : P(intro(x))\end{array}}{\vdash \mathsf{rec}(f_0, f_1)(intro(t)) = f_1(t, \lambda z{:}\mu \to A.\mathsf{rec}(f_0, f_1)(tz)) : P(intro(t))}.$$

6 General Inductive Types

As we have observed in the last section, $P_{\phi(A)}$ and $\phi^\diamond(f)$ are well-defined as long as $\phi(X)$ is of the form $\Pi x_1{:}K_1.\cdots\Pi x_m{:}K_m.X$. We will call *general inductive types* those defined in section 2 by dropping the strictness condition on positivity.

Definition 6.1 *Suppose* $\Gamma, X : \mathrm{Type}_i \vdash \phi(X) : \mathrm{Type}_i$. *(i) If* $\phi(X)$ *does not contain* X, *then it is both positive and negative with respect to* X. *(ii) If* $\phi(X) = X$, *then it is positive with respect to* X. *(iii) If* $\phi(X) = \Pi x{:}G.H$ *and* X *is positive in* H *and negative in* G, *then* X *is positive in* $\phi(X)$. *(iv) If* $\phi(X) = \Pi x{:}G.H$ *and* X *is positive in* G *and negative in* H, *then* X *is negative in* $\phi(X)$. *A* general constructor type $\Theta(X)$ *is a type of the form* $\Pi x_1{:}\phi_1(X).\cdots\Pi x_m{:}\phi_m(X).X$ *such that for each* $k \in \{1,\ldots,m\}$, *either* $\phi_k(X)$ *does not contain* X *or* $\phi_k(X) = \Pi x_{k1}{:}K_{k1}(X).\cdots\Pi x_{km_k}{:}K_{km_k}(X).X$ *with* $K_{k1}(X),\ldots,K_{km_k}(X)$ *being negative with respect to* X.

General inductive types are defined similarly as the generalized inductive types except that constructor types are replaced by general constructor types.

Some questions naturally arise:

1. Is this extra generality useful?
2. Is the meta theory of the these types anything different from that of the generalized inductive types?
3. What is the model theoretical justification?

We do not have sufficient information to answer question 1. Recently Martin Hofmann[1] discovered a programme which makes use of a non-strictly positive datatype. In Martín Abadi[2], it is shown how Scott numerals can be typed as terms of a polymorphic type which codes up a non-strict covariant functor. We need more substantial examples.

In this section we give an answer to question 3. We do this by showing that the interpretation we have given is also an interpretation of the general inductive types.

By examination, the definitions of both the model and the interpretation remain unchanged. The only places where one has to be careful are in the definition of rule set and in the interpretation of terms of introduction form. First, the definition of the rule set $\dot{R}_{\mu X.[\vec{\Theta}]}$ is unchanged. To see this, notice that in (2) $f_1,\ldots,f_m$ are all natural numbers. It follows that $\dot{R}_{\mu X.[\vec{\Theta}]}$ does define a rule set on ω. The situation is in contrast to that in Set. In (1), if $\phi_1^j(X)$ is positive but not strictly positive with respect to X then f_1 may not be in V_κ. When that is the case, $\langle n_{\theta_j}, m, f_1,\ldots,f_m\rangle$ is not in V_κ and we conclude that (1) does not define a rule set on V_κ. Secondly we need to show that the following holds:

$$\begin{array}{rl} & [\![\phi_1^j(X)]\!][X := \mathcal{P}^\alpha] \\ \preceq & [\![\phi_1^j(X)]\!][X := \mathcal{P}^{\alpha+1}], \\ \vdots & \\ & [\![\phi_m^j(X)]\!][f_1,\ldots,f_{m-1}][X := \mathcal{P}^\alpha] \\ \preceq & [\![\phi_m^j(X)]\!][f_1,\ldots,f_{m-1}][X := \mathcal{P}^{\alpha+1}]. \end{array} \tag{3}$$

We now show that positivity condition is enough to guarantee (3).

Proposition 6.2 *Suppose* $X : Type_i \vdash \phi(X) : Type_i$ *and* $A \preceq B$. *Then the following hold: (i)* $[\![\phi(X)]\!]_{X:=A} \preceq [\![\phi(X)]\!]_{X:=B}$ *if* $\phi(X)$ *is positive with respect to* X; *and dually (ii)* $[\![\phi(X)]\!]_{X:=B} \preceq [\![\phi(X)]\!]_{X:=A}$ *if* X *is negative in* $\phi(X)$. *Here* $[\![\phi(X)]\!]_{X:=A}$ *is the denotation of* $\phi(X)$ *in* Per *with* X *being interpreted as* A.

Proof: First let's mention that a judgement $x : G \vdash H : Type_i$ is interpreted as a function $[\![H]\!] : dom([\![G]\!]) \longrightarrow$Per, where $[\![G]\!]$ is a per (the denotation of G), such that if $m[\![G]\!]n$ then $[\![H]\!](m) = [\![H]\!](n)$. A term $x : G \vdash t : H$ is interpreted as a recursive function $[\![t]\!]$ such that $m[\![G]\!]n$ implies $[\![t]\!]\cdot m\{[\![H]\!](m)\}[\![t]\!]\cdot n$. The context $x : G, y : H$ is interpreted as the per $[\![x : G, y : H]\!]$ such that $m[\![x : G, y : H]\!]n$ if and only if $(m)_0[\![G]\!](n)_0$ and $(m)_1\{[\![H]\!]((m)_0)\}(n)_1$.

We now state and prove a result more general than the proposition:

Suppose $X : Type_i, x_1 : K_1,\ldots,x_n : K_n \vdash K : Type_i$. The for any

$$m \in dom([\![x_1 : K_1,\ldots,x_n : K_n]\!]_{X:=A}) \cap dom([\![x_1 : K_1,\ldots,x_n : K_n]\!]_{X:=B}),$$

$\{[\![K]\!]_{X:=A}\}(m) \preceq (\succeq)\{[\![K]\!]_{X:=B}\}(m)$ if K is positive (negative) with respect to X.

[1] Electronic forum **Types**, 18 Feb. 1993.
[2] Electronic forum **Types**, 22 Feb. 1993.

This is proved by induction on the structure of K.

- K is either X or does not contain X. Trivial.
- K is positive and $K = \Pi x{:}G.H$. Then G is negative and H positive. Because $X : \mathit{Type}, x_1 : K_1, \ldots, x_n : K_n \vdash G : \mathit{Type}$ and the induction hypothesis, for any

$$m \in dom([\![x_1 : K_1, \ldots, x_n : K_n]\!]_{X:=A}) \cap dom([\![x_1 : K_1, \ldots, x_n : K_n]\!]_{X:=B}),$$

  $\{[\![G]\!]_{X:=A}\}(m) \succeq \{[\![G]\!]_{X:=B}\}(m)$. And for $l \in \{[\![G]\!]_{X:=B}\}(m)$, $\{[\![H]\!]_{X:=A}\}(m,l) \preceq \{[\![H]\!]_{X:=B}\}(m,l)$ by induction hypothesis. It follows that $\{[\![\Pi x{:}G.H]\!]_{X:=A}\}(m) \preceq \{[\![\Pi x{:}G.H]\!]_{X:=B}\}(m)$.
- The dual to the above case is proved similarly.

The proposition is the special case when $n = 0$. □

Corollary 6.3 *The inclusion (3) holds in the case of general inductive types.*

It is worth remarking that the classical set theoretical model can not be extended to that of the general inductive types. The reason is that the κ-continuous functors are no longer available. The interpretation using rule sets ([11]) also breaks down, as we have already seen, because the negativity makes it impossible to give an inductive definition ([27]). As a consequence, the ω-SET model of the generalized inductive types given in [15] can not be modified to interpret general inductive types in the *Calculus of Constructions*.

7 Discussion

In view of the computational nature of inductive types, the result of this paper should not be surprising. Our interpretation of general inductive types shows that these types are as computational as the inductive ones. The questions about them are to do with pragmatics.

Unlike the interpretation of inductive types in SET, the requirement for set theory in our model is very preliminary . The underlying rule sets can be obtained as the least fixed points of some sequences, using the pleasant fact that $\omega \times \omega$ is an upper bound, thus avoiding the apparent impredicativity in definition 3.2. Besides the lengths of these sequences are bounded by $\aleph_1$. So the constructions of the rule sets happen in a fixed yet very small universe.

A question remains unanswered is that to what extent the result can be formulated in the language of category theory. There are at least two possible ways to look at the question: one is to use the notion of T-algebras; the other is to employ Mendler's categorical definition of recursions ([25]). The analysis in [15] suggests that neither can be tackled in a simple-minded way.

8 Acknowledgement

I would like to thank David Rydeheard whose comments on preliminary versions of this paper have been a stimulus for me to improve the readability of it, although, judging from this final version, the effort has not really paid off. Two anonymous referees have pointed out some errors. My thanks to them.

References

[1] P. Aczel. An Introduction to Inductive Definitions. In J. Barwise, editor, *Handbook of mathematical Logics*, pages 739–782. North-Holland, 1977.

[2] P. Aczel. The Type Theoretical Interpretation of Constructive Set Theory: Inductive Definitions. In P. Weingartner, R. Barcan Marcus and G.J.W. Dorn, editor, *Logic, Methodology and Philosophy of Science, VII*. North-Holland, 1986.

[3] R. Backhouse. On the Meaning and Construction of the Rules in Martin-Löf's Theory of Types. Technical Report CS 8606, Department of Mathematics and Computer Science, University of Groningen, 1986.

[4] R. Backhouse, P. Chisholm, G. Malcolm and E. Saaman. Do-it-yourself Type Theory (Part 1). *Formal Aspects of Computing*, 1:19–84, 1989.

[5] M. Beeson. Recursive Models of Constructive Set Theories. *Annals of Mathematical Logic*, 23:127–178, 1982.

[6] M. Beeson. *Foundations of Constructive Mathematics*. Springer-Verlag.

[7] R. Constable and N. Mendler. Recursive Definitions in Type Theory. In R. Parikh, editor, *Logics of Programs, LNCS 193*, pages 50–66. Springer-Verlag, 1990.

[8] T. Coquand and C. Paulin-Mohring. Inductively Defined Types. In *COLOG '88, LNCS 417*, pages 50–66. Springer-Verlag, 1990.

[9] P. Dybjer. Inductively Defined Sets in Martin-Löf's Type Theory. In *Proceedings of the Workshop on General Logic*, Edinburgh, 1988. ECS-LFCS-88-52.

[10] P. Dybjer. An Inversion Principle for Martin-Löf's Type Theory. In *Proceedings of the Workshop on Programming Logic*, Chalmer University of Technology, 1989. PMG-Report 54.

[11] P. Dybjer. Inductive Sets and Families in Martin-Löf Type Theory and Their Set-theoretic Semantics. In G. Huet and G. Plotkin, editors, *Proceedings of the First Workshop on Logical Frameworks*, pages 280–306. Cambridge University Press, 1991.

[12] P. Freyd. *Algebraic Complete Categories*, 1991.

[13] Y. Fu. *Some Semantic Issues in Type Theory*. PhD thesis, Department of Computer Science, University of Manchester, May 1992.

[14] Y. Fu. Encodings in Polymorphism, Revisited. Technical Report UMCS-93-6-4, Department of Computer Science, University of Manchester, June 1993.

[15] Y. Fu. Understanding Inductive Types in Constructions. Technical Report UMCS-93-6-5, Department of Computer Science, University of Manchester, June 1993.

[16] H. Goguen and Z. Luo. Inductive Data Types: Well-ordering Types Revisited. Technical Report, LFCS, University of Edinburgh, April 1992.

[17] G. Longo and E. Moggi. Constructive Natural Deduction and its 'Modest' Interpretation. In *Semantics of Natural and Computer Languages*, Stanford, 1990. M.I.T. Press.

[18] Z. Luo. *An Extended Calculus of Constructions*. PhD thesis, LFCS, University of Edinburgh, July 1990.

[19] Z. Luo. A Higher-order Calculus and Theory Abstraction. *Information and Computation*, 90(1):107-137, January 1991.

[20] Z. Luo. A Unifying Theory of Dependent Types: the Schematic Approach. In A. Nerode and M. Taitslin, editors, *Logical Foundations of Computer Science-Tver '92, LNCS 620*. Springer-Verlag, 1992.

[21] N. Mendler. *Inductive Definition in Type Theory*. PhD thesis, Cornell University, 1987.

[22] N. Mendler. Recursive Types and Type Constraints in Second-order Lambda Calculus. In *Proceedings of the Second Symposium on Logic in Computer Science*, pages 30–36. IEEE Computer Science Press, June 1987.

[23] N. Mendler. *Inductive Types via Initial Algebras, in Martin-Löf Type Theory*, July 1990.

[24] N. Mendler. Inductive Types and Type Constraints in the Second-order Lambda Calculus. *Annals of Pure and Applied Logic*, 51:159-172, 1991.

[25] N. Mendler. Predicative Type Universes and Primitive Recursion. In *Proceedings of the Sixth Symposium on Logic in Computer Science*, pages 173-184. IEEE Computer Science Press, July 1991.

[26] P. Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Napels, 1984.

[27] Y. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, 1974.

[28] B. Nordström, K. Peterson and J. Smith. *Programming in Martin-Löf's Type Theory–an introduction*, volume 7 of *International series of monographs on computer science*. Oxford University Press, 1990.

[29] C. Ore. The Extended Calculus of Constructions (*ECC*) with Inductive Types. *Information and Computation*, 99:231–164, August 1992.

[30] F. Pfenning and C. Paulin-Mohring. Inductively Defined Types in the Calculus of Constructions. In M. Main *et al.*, editor, *Mathematical Foundations of Programming Semantics, LNCS442*, pages 209–228. Springer-Verlag, 1991.

[31] B. Smith and G. Plotkin. The Categorical Solution of Recursive Domain Equation. *SIAM Computing*, 11(4):761–783, 1982.

[32] G. Wraith. A Note on Categorical Datatypes. In D. Pitt, D. Rydeheard, P. Dybjer, A. Pitts and A. Poigné, editors, *Category Theory and Computer Science, LNCS 389*, pages 213–223, Manchester, September 1989. Springer-Verlag.