Computer Science
University of Manchester

Recursive Models of General Inductive Types

Yuxi Fu

Technical Report UMCS–93–6–6

Recursive Models of General Inductive Types

Yuxi Fu

Department of Computer Science
University of Manchester
Oxford Rd., Manchester, U.K.

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.

Technical Reports issued by the Department of Computer Science, Manchester University, are available by anonymous ftp from `m1.cs.man.ac.uk` (130.88.13.4) in the directory `/pub/TR`. The files are stored as PostScript, in compressed form, with the report number as filename. Alternatively, reports are available by post from The Computer Library, Department of Computer Science, The University, Oxford Road, Manchester M13 9PL, U.K.

Interest in inductively defined types has been around for some time. An early work on a categorical approach to recursively defined types is in [SP82], 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 [Men87a, Men87b, Men91a], see also [CM85], 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 [BCMS89, Dyb89, Dyb91] and in [CPM90, Luo92]. 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 [Wra89, PPM91, GL92, Fu93]. The argument against this approach is threefold. First, if we think of type theory as a framework in which computational objects are defined and manipulated, then the correct notion of equality seems to be the intensional one. When formulated appropriately, such intensional equality is internally extensional as shown in [Luo92]. Second, the meta-theory of the extensional theory 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 [Dyb88] 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 [Ore92]. 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 [Men90, Men91b]. Related to this is the work [Fre91] 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 [CPM90]. Unfortunately, a monotonic and κ-bounded functor on the category ω-SET of ω-sets ([Luo90, Ore92]) 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 ([Acz86, Dyb91]). This approach, which is mathematically weaker, does generalize to ω-SET. For more details on both the subjects, see [Fu92a, Fu92b].

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 5.

1 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. Intuitively the small universes satisfy the following properties:

$$\begin{array}{l} Type_0 \in Type_1 \in Type_2 \in \cdots, \\ Type_0 \subset Type_1 \subset Type_2 \subset \cdots. \end{array}$$

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

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

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

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

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

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

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

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

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

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

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

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

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 [CPM90] to get things off the ground. We will call this language ML^{∞}_{gi}.

Definition 1.1 *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* $\phi(X) = \Pi x_1{:}K_1. \cdots \Pi x_m{:}K_m.X$ $(m \in \omega)$ *such that* X *does not occur in any of* $K_1, \cdots, K_m$.

If $\Gamma \vdash A : Type_i$ and $\Gamma \vdash P_A : A \rightarrow Type_i$, 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, then we can extend P_A to $\Gamma \vdash P_{\phi(A)} : \phi(A) \rightarrow Type_i$, 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 for every inhabitant of A. We extend f to a proof $\phi^\diamond(f)$ showing that $P_{\phi(A)}$ holds for all members of $\phi(A)$, that is

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

in the same fashion:

- If $\phi(X) = X$, $\phi^\diamond(f) \stackrel{\text{def}}{=} f$.
- 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 1.2 *Suppose* $\Gamma, X : Type_i \vdash \Theta(X) : Type_i$. *We say that* Θ *is a* constructor *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.*

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 for each $k \in [1\cdot\cdot n]$.

Formation

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

where $\Theta_1(X), \ldots, \Theta_n(X)$ are constructors. 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 $k \in [1\cdot\cdot n]$.

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 $k \in [1\cdot\cdot n]$, $\phi_{kk_1}(X), \ldots, \phi_{kk_p}(X)$ 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

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

for $1 \le k \le n$. 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

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

Example 1.3 The W-types ([ML84, NPS90]). This example is meant to bring out the intuition behind the above rules. There is only one constructor: $\Theta = \Pi x{:}A.\Pi y{:}B(x) \rightarrow X.X$. Here $\phi_1(X) = A$ and $\phi_2(X) = B(x) \rightarrow X$. The introduction rule is the familiar one:

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

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

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

The generalized inductive types are two dimensional generalizations of W-types. ■

2 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 [Bee82, Bee85]. 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.

2.1 Some Preliminaries

Definition 2.1 *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 $x \in \{n|nAn\}$. 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 $\pi_0 m A \pi_0 n$ and $\pi_1 m B \pi_1 n$. 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 $\pi_0 m = \pi_0 n = 0$ and $\pi_1 m A \pi_1 n$ or $\pi_0 m = \pi_0 n = 1$ and $\pi_1 m B \pi_1 n$. 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 2.2 *([Acz77]) 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 $\mathcal{I}(R) \stackrel{\text{def}}{=} \bigcap\{A|A$ is R-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 [Acz86, Dyb91]. 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 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.

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, where $\phi_i^j(X) = \Pi x_{i1} : K_{i1}^j. \cdots \Pi x_{im_i} : K_{im_i}^j.X$ $(m \in \omega)$ for $i \in [1 \cdots m]$. The rule sets R_{Θ_j} is defined as the following set:

$$\left\{ \frac{\cup_{k=1\cdots m} range(f_k)}{\langle n_{\Theta_j}, m, f_1, \ldots, f_m\rangle} \;\middle|\; \begin{array}{l} f_1 \in \pi([\![K_{11}^j]\!], \ldots, \pi([\![K_{1m_1}^j]\!], V_\kappa)\cdots) \wedge \cdots \wedge \\ f_m \in \pi([\![K_{m1}^j]\!][f_1,\ldots,f_{m-1}], \ldots, \pi([\![K_{mm_m}^j]\!][f_1,\ldots,f_{m-1}], V_\kappa)\cdots) \end{array} \right\}$$

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 $\cup_{j=1\cdots n} R_{\Theta_j}$. 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. Two new pers are defined as follows:

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

In the PER interpretation, a context is interpreted as a per. A type $\Gamma \vdash A$ is modelled 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 $\pi([\![A]\!](\alpha,\_),[\![B]\!](\alpha,\_))$.

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 $\omega\times\omega$.

$$\dot{R}_{\Theta_j} \stackrel{\text{def}}{=} \left\{ \frac{\cup_{k=1\cdots m} range(f_k)}{\langle n_{\Theta_j}, m, f_1, \ldots, f_m\rangle} \;\middle|\; \begin{array}{l} f_1 \in \pi([\![K_{11}^j]\!], \ldots, \pi([\![K_{1m_1}^j]\!], \omega\times\omega)\cdots) \wedge \cdots \wedge \\ f_m \in \pi([\![K_{m1}^j]\!][f_1,\ldots,f_{m-1}], \ldots, \pi([\![K_{mm_m}^j]\!][f_1,\ldots,f_{m-1}], \omega\times\omega)\cdots) \end{array} \right\}$$

Let $\dot{R}_{\mu X.[\vec{\Theta}]} \stackrel{\text{def}}{=} \cup_{i=1\cdots n} \dot{R}_{\Theta_i}$. $\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})$ is a set, while 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 $\varnothing$.

- $\mathcal{P}^{\alpha+1} \stackrel{\text{def}}{=} \cup_{j=1\cdots n} S_j$ where S_j is the following set:

$$\left\{ \begin{array}{l|l} & \langle n_{\Theta_j}, m, f_1, \ldots, f_m\rangle \in \mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}) \wedge \\ \langle\langle n_{\Theta_j}, m, f_1, \ldots, f_m\rangle, & \langle n_{\Theta_j}, m, g_1, \ldots, g_m\rangle \in \mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]}) \wedge \\ \langle n_{\Theta_j}, m, g_1, \ldots, g_m\rangle\rangle & f_1\{\pi([\![K^j_{11}]\!], \ldots, \pi([\![K^j_{1m_1}]\!], \mathcal{P}^\alpha)\cdots)\} g_1 \wedge \cdots \wedge \\ & f_m\{\pi([\![K^j_{m1}]\!][f_1, \ldots, f_{m-1}], \ldots, \pi([\![K^j_{mm_m}]\!][f_1, \ldots, f_{m-1}], \mathcal{P}^\alpha)\cdots)\} g_m \end{array} \right\}.$$

  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:

$$\frac{\vdash t_1 : \phi^j_1(\mu X.[\vec{\Theta}]), \ \cdots \ \vdash t_m : \phi^j_m[t_1, \ldots, t_{m-1}/x_1, \ldots, x_{m-1}](\mu X.[\vec{\Theta}])}{\vdash \mathsf{intro}^\mu_j(t_1, \ldots, t_m) : \mu X.[\Theta_1, \ldots, \Theta_n]}.$$

By induction hypothesis, we have

$$\begin{array}{rcl} [\![t_1]\!] & \in & \pi([\![K^j_{11}]\!], \ldots, \pi([\![K^j_{1m_1}]\!], \mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})))\cdots), \\ & \vdots & \\ [\![t_m]\!] & \in & \pi([\![K^j_{m1}]\!][[\![t_1]\!], \ldots, [\![t_{m-1}]\!]], \ldots, \pi([\![K^j_{mm_m}]\!][[\![t_1]\!], \ldots, [\![t_{m-1}]\!]], \mathcal{P}(\mathcal{I}(\dot{R}_{\mu X.[\vec{\Theta}]})))\cdots). \end{array}$$

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

2.2 The Model

The purpose of this section is to construct a model $\mathcal{M}$ on the partial applicative structure $(\omega, \cdot)$. In section 3 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 definition, one first postulates a name, one then attaches a subset of ω to the name, finally one defines an equivalence relation on the subset.* 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. We also assume a standard Gödel numbering φ of n-ary functions for each $n \geq 1$. $\varphi_f(a_1, \ldots, a_n)$ will be simplified to $f \cdot (a_1, \ldots, a_n)$. 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
$$\begin{array}{l}\forall x(\mathcal{M}_0 \models x \in ext(a) \Rightarrow \mathcal{M}_0 \models n\cdot x \in ext(b\cdot x))\wedge\\ \forall x, y(\mathcal{M}_0 \models x = y \in ext(a) \Rightarrow \mathcal{M}_0 \models n\cdot x = n\cdot y \in ext(b\cdot x))\end{array}$$
and $\mathcal{M}_0 \models m = n \in ext(\pi(a,b))$ iff
$$\begin{array}{l}\mathcal{M}_0 \models m \in ext(\pi(a,b))\wedge\mathcal{M}_0 \models n \in ext(\pi(a,b))\wedge\\ \forall x(\mathcal{M}_0 \models x \in ext(a) \Rightarrow \mathcal{M}_0 \models m\cdot x = n\cdot x \in ext(b\cdot x)).\end{array}$$

4. Suppose $c^1, c^2, \cdots, c^n$ are names already defined. Let's abbreviate $\langle 2, n, \langle n_1, c^1\rangle, \cdots, \langle n_n, c^n\rangle\rangle$ to ind. Define
$$\mathcal{M}_0 \models ind, \quad \mathcal{M}_0 \models ext(ind) = \mathcal{P}(\mathcal{I}(R_{ind}))$$
where $R_{ind} = \cup_{k=1\cdot\cdot n} R_{\langle n_k, c^k\rangle}$. $R_{\langle m,d\rangle}$ 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):
$$R_{\langle m,d\rangle} \stackrel{\text{def}}{=} \left\{ \frac{\cup_{k=1\cdot\cdot m} range(t_i, (d^i)_1)}{\langle intro, m, t_1, \cdots, t_m\rangle} \;\middle|\; \begin{array}{l}(d^1)_0 = 1 \wedge \mathcal{M}_0 \models t_1 \in ext((d^1)_1)\wedge\\ (d^2)_0 = 1 \wedge \mathcal{M}_0 \models t_2 \in ext((d^2)_1)\\ \wedge\cdots\wedge\\ (d^m)_0 = 1 \wedge \mathcal{M}_0 \models t_m \in ext((d^m)_1)\end{array}\right\}$$
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
$$range(t,d) \stackrel{\text{def}}{=} \left\{ t\cdot a_1\cdots a_n \;\middle|\; \begin{array}{l} n\in\omega\wedge(d)_0 = 1\wedge\mathcal{M}_0 \models a_1 \in ext((d)_1)\wedge\\ ((d)_2\cdot a_1)_0 = 1 \wedge \mathcal{M}_0 \models a_2 \in ext(((d)_2\cdot a_1)_1)\\ \wedge\cdots\wedge((\cdots((d)_2\cdot a_1)_2\cdot a_2\cdots)_2\cdot a_{n-1})_0 = 1\\ \wedge\mathcal{M}_0 \models a_n \in ext(((\cdots((d)_2\cdot a_1)_2\cdot a_2\cdots)_2\cdot a_{n-1})_1)\\ \wedge((\cdots((d)_2\cdot a_1)_2\cdot a_2\cdots)_2\cdot a_{n-1})_2\cdot a_n = \langle 0,1\rangle\end{array}\right\}$$
provided that $\mathcal{M}_0 \models t \in ext(d)$. Notice that when $n = 0$, $t\cdot a_1\cdots a_n$ is just t. As the set of all conclusions of $\cup_{k=1\cdot\cdot n} range(n_k, c^k)$ 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 $k \in [1\cdots n]$.

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 $\gamma(a,b) \stackrel{\text{def}}{=} \langle 4, a, b\rangle$.
$$\mathcal{M} \models (c^0, c^1) \in ext(\gamma(a,b)) \Leftrightarrow \mathcal{M} \models c^0 \in ext(a) \wedge \mathcal{M} \models c^1 \in ext(b \cdot c^0),$$
$$\mathcal{M} \models (c^0, c^1) = (d^0, d^1) \in ext(\gamma(a,b)) \Leftrightarrow \mathcal{M} \models c^0 = d^0 \in ext(a) \wedge \mathcal{M} \models c^1 = d^1 \in ext(bc^0).$$
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 $\mathcal{M} \models \gamma(\gamma(a,b),f)$ with $len(\gamma(\gamma(a,b),f)) = len(\gamma(a,b)) + 1$.
$$\begin{aligned} & \mathcal{M} \models (c^0,\ldots,c^{n-1},c^n) \in ext(\gamma(\gamma(a,b),f)) \\ \Leftrightarrow\ & \mathcal{M} \models (c^0,\ldots,c^{n-1}) \in ext(\gamma(a,b)) \wedge \mathcal{M}_i \models c^n \in ext(f \cdot (c^0,\ldots,c^{n-1})), \\ & \mathcal{M} \models (c^0,\ldots,c^{n-1},c^n) = (d^0,\ldots,d^{n-1},d^n) \in ext(\gamma(\gamma(a,b),f)) \\ \Leftrightarrow\ & \mathcal{M} \models (c^0,\ldots,c^{n-1}) = (d^0,\ldots,d^{n-1}) \in ext(\gamma(a,b)) \wedge \mathcal{M}_i \models c^n = d^n \in ext(f \cdot (c^0,\ldots,c^{n-1})). \end{aligned}$$
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 $\mathcal{M}$.

3 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 $A\hat{}$ 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 $\langle 0, 1\rangle$.

- **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 over $\langle 0, 1\rangle$.

- **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 $\Lambda x_1, \ldots, x_n.x_i$.

- **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 $\widehat{\lambda x{:}A.M}$ be this g. We need to show that $\mathcal{M} \models \widehat{\lambda x{:}A.M} : \hat{\Gamma} \Longrightarrow \widehat{\Pi x{:}A.B}$. 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 $\widehat{\lambda x{:}A.M}\cdot a = g(\hat{M}, a^0, \ldots, a^{n-1})\cdot a = \hat{M}\cdot(a^0, \ldots, a^{n-1}, a)$. 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 $\mathcal{M}_i \models \widehat{\lambda x{:}A.M}\cdot a \in ext(h(\hat{B}, a^0, \ldots, a^{n-1})\cdot a)$ where h is obtained in the previous case. The second condition in the definition of γ can be similarly verified. Therefore $\mathcal{M} \models \widehat{\lambda x{:}A.M} : \hat{\Gamma} \Longrightarrow \widehat{\Pi x{:}A.B}$.

- **Application.** By induction hypotheses, $\mathcal{M} \models \hat{M} : \hat{\Gamma} \Longrightarrow \widehat{\Pi x{:}A.B}$ and $\mathcal{M} \models \hat{N} : \hat{\Gamma} \Longrightarrow \hat{A}$. $\widehat{M\cdot N}$ is interpreted as $\Lambda x_1, \ldots, x_n.(\hat{M}\cdot(x_1, \ldots, x_n))\cdot(\hat{N}\cdot(x_1, \ldots, x_n))$.

- **Substitution.** Suppose $\Gamma \vdash N : A$ and $\Gamma, x : A \vdash M : B$. Then $\Gamma \vdash M[N/x] : B[N/x]$ is interpreted by $\mathcal{M} \models \hat{M} \circ \langle id, \hat{N}\rangle \in ext(\gamma(\hat{\Gamma}, \widehat{B[N/x]}))$ 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 $\mathcal{M} \models \hat{B} \circ \langle id, \hat{N}\rangle : \hat{\Gamma} \Longrightarrow \Lambda X.\langle 3, i + 1\rangle$ 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 $\mathcal{M} \models \hat{M} : \gamma(\hat{\Gamma}, \hat{A}) \Longrightarrow \hat{B}$.

- **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 all $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$. We assign to $\hat{\mu}_{(a^0,\ldots,a^{n-1})}$ the number

  $$\langle 2, n, \langle n_1, \widehat{\Theta_1}\cdot(a^0, \ldots, a^{n-1}, \langle 0,1\rangle)\rangle, \ldots, \langle n_n, \widehat{\Theta_n}\cdot(a^0, \ldots, a^{n-1}, \langle 0,1\rangle)\rangle\rangle.$$

  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 every $k \in [1\cdots n]$, and any $\mathcal{M} \models (a^0, \ldots, a^{n-1}) \in ext(\hat{\Gamma})$,

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

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

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

  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 start 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

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

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

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

  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 $k\in[1\cdot\cdot n]$. Then

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

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

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

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

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 $(g\cdot(a^0,\ldots,a^{n-1}))\cdot(\hat{N}\cdot(a^0,\ldots,a^{n-1})) = ((\lambda x{:}A.M)\cdot N)\hat{}\cdot(a^0,\ldots,a^{n-1})$.

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, $\mathcal{M}\models\hat{f}:\hat{\Gamma}\Longrightarrow\Pi x\hat{:}A.B$. 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 $(\hat{f}\cdot(a^0,\ldots,a^{n-1}))\cdot a^n$. It follows from the denotation of $\mathcal{M}_i$ 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 $\mathcal{M}\models g=\hat{f}:\hat{\Gamma}\Longrightarrow\widehat{\Pi x{:}A.B}$.

4 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

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

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

$$\frac{}{\vdash \perp : \mu} \qquad \frac{\vdash t : (\mu \to A) \to \mu}{\vdash intro(t) : \mu} \qquad \frac{\vdash P : \mu \to Type_i \quad \vdash f_0 : P(\perp)}{\vdash \mathsf{rec}(f_0, f_1)(\perp) = f_0 : P(\perp)}$$

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

5 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 1 by dropping the strictness conditions on positivity.

Definition 5.1 *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 *occurs positively in* H *and negatively in* G*, then* X *occurs positively in* $\phi(X)$*. (iv) If* $\phi(X) = \Pi x{:}G.H$ *and* X *occurs positively in* G *and negatively in* H*, then* X *occurs negatively in* $\phi(X)$*.*

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.

As the author understands, some researchers are currently investigating problems related to 2.

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 place where one has to be careful is in the interpretation of terms of introduction form. Specifically we want the following to hold:

$$\begin{array}{ll} & \pi([\![K^j_{11}]\!][\mathcal{P}^\alpha], \ldots, \pi([\![K^j_{1m_1}]\!][\mathcal{P}^\alpha], \mathcal{P}^\alpha)\cdots) \\ \preceq & \pi([\![K^j_{11}]\!][\mathcal{P}^{\alpha+1}], \ldots, \pi([\![K^j_{1m_1}]\!][\mathcal{P}^{\alpha+1}], \mathcal{P}^{\alpha+1})\cdots) \\ \vdots & \\ & \pi([\![K^j_{m1}]\!][\mathcal{P}^\alpha][f_1, \ldots, f_{m-1}], \ldots, \pi([\![K^j_{mm_m}]\!][\mathcal{P}^\alpha][f_1, \ldots, f_{m-1}], \mathcal{P}^\alpha)\cdots) \\ \preceq & \pi([\![K^j_{m1}]\!][\mathcal{P}^{\alpha+1}][f_1, \ldots, f_{m-1}], \ldots, \pi([\![K^j_{mm_m}]\!][\mathcal{P}^{\alpha+1}][f_1, \ldots, f_{m-1}], \mathcal{P}^{\alpha+1})\cdots) \end{array} \tag{1}$$

We now show that positivity condition is enough to guarantee (1).

Proposition 5.2 *Suppose $X : \mathrm{Type} \vdash \phi(X) : \mathrm{Type}$ and $A \preceq B$. Then (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 appears negatively 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 $x : G \vdash H : \mathit{Type}$ is interpreted as a function $[\![H]\!] : \mathit{dom}([\![G]\!]) \longrightarrow \mathrm{PER}$, 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 $\pi_1 m[\![G]\!]\pi_1 n$ and $\pi_2 m\{[\![H]\!](\pi_1 m)\}\pi_2 n$.

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

> Suppose $X : \mathit{Type}, x_1 : K_1, \ldots, x_n : K_n \vdash K : \mathit{Type}$. The for any
>
> $m \in \mathit{dom}([\![x_1 : K_1, \ldots, x_n : K_n]\!]_{X:=A}) \cap \mathit{dom}(([\![x_1 : K_1, \ldots, x_n : K_n]\!]_{X:=B})$,
>
> $\{([\![K]\!]_{X:=A}\}(m) \preceq (\succeq)\{([\![K]\!]_{X:=B}\}(m)$ 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 complexity 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

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

  $\{[\![G]\!]_{X:=A}\}(m) \succeq \{[\![G]\!]_{X:=B}\}(m)$. 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 5.3 *The inclusion (1) 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 ([Dyb91]) also breaks down because the negativity makes it impossible to give an inductive definition ([Mos74]). As a consequence, the ω-SET model of the generalized inductive types given in [Fu92b] can not be modified to interpret general inductive types in the *Calculus of Constructions*.

6 Acknowledgement

The material in this paper is taken from the author's Ph.D thesis ([Fu92a]). My supervisor David Rydeheard has been very helpful all the way over the last two years. The examiners Peter Aczel and Martin Hyland have raised some interesting points and made quite a few comments.

The financial suport from Yu-Gang Bau, the British Council, the State Education Commission of China is gratefully acknowledged. During the final stage of the preparation of this paper, the author is funded under the CLICS-II project.

References

[Acz77] P. Aczel. An introduction to inductive definitions. In J. Barwise, editor, *Handbook of Mathematical Logics*, pages 739–782. North-Holland, 1977.

[Acz86] P. Aczel. The type theoretic interpretation of constructive set theory: Inductive definitions. In P. Weingartner R. Barcan Marcus, G.J.W. Dorn, editor, *Logic, Methodology and Philosophy of Science, VII*. North-Holland, 1986.

[BCMS89] R. Backhouse, P. Chisholm, G. Malcolm, and E. Saaman. Do-it-yourself type theory (part 1). *Formal Aspects of Computing*, 1:19–84, 1989.

[Bee82] M. Beeson. Recursive models of constructive set theories. *Annals of Mathematical Logic*, 23:127–178, 1982.

[Bee85] M. Beeson. *Foundations of Constructive Mathematics*. Springer-Verlag, 1985.

[CM85] R. Constable and N. Mendler. Recursive definitions in type theory. In R. Parikh, editor, *Logics of Programs, LNCS193*, pages 61–78. Springer-Verlag, 1985.

[CPM90] T. Coquand and C. Paulin-Mohring. Inductively defined types. In *COLOG '88, LNCS 417*, pages 50–66. Springer-Verlag, 1990.

[Dyb88] P. Dybjer. Inductively defined sets in Martin-Löf's set theory. In *Proceedings of the Workshop on General Logic*, Edinburgh, 1988. ECS-LFCS-88-52.

[Dyb89] 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.

[Dyb91] 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.

[Fre91] P. Freyd. *Algebraic Complete Categories*, 1991.

[Fu92a] Yuxi Fu. *Some Semantic Issues In Type Theory*. PhD thesis, Department of Computer Science, University of Manchester, May 1992.

[Fu92b] Yuxi Fu. *Understanding Inductive Types in Constructions*. Department of Computer Science, University of Manchester, August 1992.

[Fu93] Yuxi Fu. *Encodings in Polymorphism, Revisited*. Department of Computer Science, University of Manchester, January 1993.

[GL92] H. Goguen and Z. Luo. Inductive data types: Well-ordering types revisited. Technical report, LFCS, University of Edinburgh, April 1992.

[Luo90] Z. Luo. *An Extended Calculus of Constructions*. PhD thesis, LFCS, University of Edinburgh, Edinburgh, July 1990.

[Luo92] 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, 1992.

[Men87a] N. Mendler. *Inductive Definition in Type Theory.* PhD thesis, Cornell University, 1987.

[Men87b] 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.

[Men90] N. Mendler. *Inductive Types via Initial Algebras, in Martin-Löf Type Theory*, July 1990.

[Men91a] N. Mendler. Inductive types and type constraints in the second-order lambda calculus. *Annals of Pure and Applied Logic*, 51:159–172, 1991.

[Men91b] 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.

[ML84] P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, Napels, 1984.

[Mos74] Y. Moschovakis. *Elementary Induction on Abstract Structures.* North-Holland, 1974.

[NPS90] B. Nordström, K. Petersson, 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, Oxford, 1990.

[Ore92] C. Ore. The extended calculus of constructions (ECC) with inductive types. *Information and Computation*, 99:231–264, August 1992.

[PPM91] 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.

[SP82] B. Smith and G. Plotkin. The categorical solution of recursive domain equation. *SIAM Computing*, 11(4):761–783, 1982.

[Wra89] G. Wraith. A note on categorical datatypes. In D. Pitt, Rydeheard, P. Dybier, A. Pitts, and A. Poigné, editors, *Category Theory and Computer Science, LNCS 389*, pages 213–223, Manchester, September 1989. Springer Verlag.