Computer Science
University of Manchester

Understanding Inductive Types in Constructions

Yuxi Fu

Technical Report UMCS–93–6–5

Understanding Inductive Types in Constructions

Yuxi Fu

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

Abstract

In this paper we extend the *Calculus of Constructions* with generalized inductive types. The extension is justified by showing that the usual set theoretical model can be effectivized. It is also pointed out that the model given in a published paper for a collection of inductive types in a different style is wrong.

1 Introduction

Constructive type theory has been developed around three principles. The oft-quoted *Curry-Howard's principle* ([How80]) may be stated as:

Constructive propositions are types.

This is what underlies Girard's work on system $\mathcal{F}$ and some recent interest in polymorphism and functional programming languages. Here the notion of sets is missing, which is the prime reason for the limitation of the languages designed on this principle. Perhaps this is not a serious drawback as far as functional programming is concerned. The *Martin-Löf's principle* is on the other end of the pole. It says:

Constructive sets are types.

What is lacking in these calculi is a logic[1] As a consequence the Leibniz equality is not expressible. We are therefore forced to internalize the definitional equality. Hence the somewhat odd *identity sets*. Languages in this group have good mechanism for defining and manipulating data types. But applications to program specifications and verifications are limited due to the absence of a logic. Since [Coq86], and especially [Luo91] (see also [Luo90a]), we now know that it is perfectly sound to unify those two kinds of languages. I will call *Russell's principle* the following:

The range of a constructive significance is a type.

The idea has its origin in *Principia Mathematica* ([RW12]). To make sensible use of this principle, one has first of all to answer the question: what is the relationship between propositions and sets? Obviously things should be structured. By Russell's principle, there should be a type Prop of all propositions. There are three obvious choices. *1.* The sets sit on top of Prop ($\mathsf{Prop} : Type_0$), this is the *Calculus of Constructions* ([CH88, Coq86, Coq90, Luo90a]). *2.* The sets sit within Prop, ($U_\infty \subset \mathsf{Prop}$), the *Calculus of Computation.* *3.* The sets and Prop are separated, the *Calculus of separated Constructions.* The second language has elimination rules for closed universes, but the usefulness of it is questionable. For a detailed description of the third language, see [Fu93c]. **ECC** is the best representative in the *CC*-like calculi. Part of the expressive power of **ECC** comes from the fact that $Type_i$ ($i \in \omega$) are open universes, which is a consequence of $\mathsf{Prop} : Type_0$. The studies carried out in [Ber89] and [Luo90b] clarify, both negatively and positively, the role of propositions in *Constructions.* In the first paper, a counter example is given to show that if we identify sets with propositions, the *Calculus of Constructions* is not conservative over HOL, the constructive version of Church's higher order logic ([Chu40]). In the second one, it is shown that, when properly separated, it

[1] To my way of thinking, there is no such thing as a "ramified logic". Logics are intrinsically impredicative, while sets must be stratified.

is conservative over HOL. The significance of their work is to show that the first two principles are different.

There remain many questions about *Constructions*. To start with, there is no civilized way of carrying out metatheoretical investigations into various *Enriched Calculi of Constructions*. The model theoretic methods such as the "candidats de reductibilité" are doomed to be complicated. The main idea in those methods is to reduce the issue to a proof of the soundness of some term model, which inherits all the combinatorial complexity from a case-study approach. Second we don't know the expressive power of such languages. What we now can do is to interpret various traditional constructive languages in **ECC** to have a good understanding of the relationships. In doing so, one should take into account of the fact that **ECC** contains a logic. The usual interpretation in Martin-Löf's set theory, see for example [Acz86], might have a more faithful counterpart in **ECC**—that is logical formulas are interpreted as (impredicative) propositions while the notion of sets is kept at the *Type*-level as it were. See [Fu93c] for an initial attempt.

Russell's principle subsumes the other two. As a result, we have this useful slogan:

> One can extend the *Calculus of Constructions* to include whatever one has in Martin-Löf's set theory.

Two points need be clarified. First, by Martin-Löf's set theory we mean the traditional one. What has prevented Martin-Löf from calling something type theory should be a good enough reason for us not to call it Martin-Löf's set theory. Second, one might argue against the slogan by saying that universes in Martin-Löf's set theory are closed while $Type_i$ ($i \in \omega$) are not. This is really a misconception. The closed universes should sit inside each $Type_i$. The *Unifying Theory of Dependent Types* [GL92] is a calculus proposed along this line, although the author's emphasis is on the decidability of the language.

The main purpose of this paper is to reinforce our confidence in the above slogan by taking a look at what happens in the model theory. In section 2 we study the semantics of the *Calculus of Constructions* extended with the generalized inductive types in a constructive setting. By transplanting the set-theoretical models to the category ω-SET, we interpret various enriched *Calculus of Constructions*. In section 3, we set up a general categorical framework and prove a general categorical result. We show how **ECC** extended with another class of inductive types can be interpreted in ω-SET. This general categorical results then motivates the formulation of another class of inductive types.

2 The Traditional Approach

In [Dyb91], a set-theoretic model is given to the inductive types defined in Martin-Löf's set theory, using Aczel's notion of rule sets ([Acz77]). A similar model using κ-continuous functors is described in [CPM90]. Neither of them can treat inductive types in *Constructions*, which is impredicative. In [Dyb91] the author says, we quote, "This interpretation does of course *not*[2] extend to the full system of the calculus of constructions extended with inductive types. Having a set-theoretic model is one of the properties which distinguish predicative from the impredicative type theory". Of course nothing prevents us from giving a *constructive* set theoretical model. In fact, the model is one version of the *recursive* set theory. This reflects the computational aspect of the inductive types. Inductive types *are* by essence computational objects. In this section we are going to transplant the classical set-theoretical construction to the category ω-SET. The definition of the interpretation follows [Dyb91] closely. Our contribution is to effectivize the sets obtained from certain rule sets.

2.1 The Enriched Calculus of Constructions

We present the language in the style of Tarski. This formulation ([Coq90, Luo90b]) has some theoretical advantages. For instance, the standard ω-SET model is perfect for this explicit calculus. The following rules are the skeleton for the *Calculus of Constructions.*

$$\frac{}{\vdash Prop : Type} \qquad \frac{\Gamma \vdash A : Type}{\Gamma, x : A \vdash Prop : Type} \qquad \frac{\Gamma, x : A, \Gamma' \vdash Prop : Type}{\Gamma, x : A, \Gamma' \vdash x : A}$$

$$\frac{\Gamma \vdash P : Prop}{\Gamma \vdash Prf(P) : Type} \qquad \frac{\Gamma, x : A \vdash b : B}{\Gamma \vdash \lambda x{:}A.b : \Pi x{:}A.B} \qquad \frac{\Gamma \vdash f : \Pi x{:}A.B \quad \Gamma \vdash a : A}{\Gamma \vdash fa : B[a]}$$

$$\frac{\Gamma, x : A \vdash B : Type \quad \Gamma \vdash A : Type}{\Gamma \vdash \Pi x{:}A.B : Type} \qquad \frac{\Gamma, x : A \vdash P : Prop}{\Gamma \vdash \forall x{:}A.P : Prop}$$

$$\frac{\Gamma, x : A \vdash M : Prf(P)}{\Gamma \vdash \Lambda x{:}A.M : Prf(\forall x : A.P)} \qquad \frac{\Gamma \vdash M : Prf(\forall x : A.P) \quad \Gamma \vdash N : A}{\Gamma \vdash M \bullet N : Prf(P[N/x])}$$

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash B : Type \quad A \simeq B}{\Gamma \vdash a : B}$$

We have omitted all the nonextensional equational rules.

There are many ways to extend the language just described. Here we enrich it with the generalized inductive types as formulated in [Dyb91].

Context

$$\frac{\Gamma \vdash A_1 : Type, \quad \Gamma, x_1 : A_1 \vdash A_2 : Type, \ldots, \quad \Gamma, x_1 : A_1, \ldots, x_{n-1} : A_{n-1} \vdash A_n : Type}{\Gamma \vdash [x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n] \ Ctxt}$$

Context realization

$$\frac{\Gamma \vdash a_1 : A_1, \quad \Gamma \vdash a_2 : A_2[a_1], \ldots, \Gamma \vdash a_n : A_n[a_1, \ldots, a_{n-1}]}{\Gamma \vdash (a_1, \ldots, a_n) : [x_1 : A_1, x_2 : A_2, \ldots, x_n : A_n]}$$

[2]Emphasized by the present author.

For a context $\Delta \stackrel{\text{def}}{=} [x_1 : A_1, \ldots, x_n : A_n]$, $\Pi z : \Delta.C$ is intuitively 'isomorphic' to $\Pi x_1 : A_1. \cdots \Pi x_n : A_n.C$.[3] Substitution into contexts is computed componentwisely: $\Delta[a]$ means $[x_1 : A_1[a], \ldots, x_n : A_n[a]]$.

Formation

$$\frac{\begin{array}{l}\Gamma \vdash \Delta_1\ Ctxt,\ \ \Gamma\Delta_1 \vdash \Phi_{11}\ Ctxt, \ldots,\ \Gamma\Delta_1 \vdash \Phi_{1n_1}\ Ctxt, \ldots, \\ \Gamma \vdash \Delta_n\ Ctxt,\ \ \Gamma\Delta_n \vdash \Phi_{n1}\ Ctxt, \ldots, \Gamma\Delta_n \vdash \Phi_{nn_n}\ Ctxt\end{array}}{\Gamma \vdash \mu(\Delta_1, \ldots, \Delta_n).[\Phi_{11}, \ldots, \Phi_{1n_1}; \ldots; \Phi_{n1}, \ldots, \Phi_{nn_n}] : Type}$$

We will often abbreviate $\mu(\Delta_1, \ldots, \Delta_n).[\Phi_{11}, \ldots, \Phi_{1n_1}; \ldots; \Phi_{n1}, \ldots, \Phi_{nn_n}]$ to $\mu(\vec{\Delta}).[\vec{\Phi}]$ or even to μ.

Introduction

$$\frac{\Gamma \vdash \vec{a} : \Delta_i,\ \ \Gamma \vdash b_k : \Phi_{ik}[\vec{a}] \to \mu(\vec{\Delta}).[\vec{\Phi}],\ k \in [1 \cdot\cdot n_i]}{\Gamma \vdash \mathsf{intro}_i^\mu(\vec{a}, b_1, \ldots, b_{n_i}) : \mu(\vec{\Delta}).[\vec{\Phi}]}(i = 1 \cdot\cdot n)$$

Elimination

$$\frac{\begin{array}{l}\Gamma, z : \mu(\vec{\Delta}).[\vec{\Phi}] \vdash C : Type \\ \Gamma \vdash d_i \in \Pi\vec{x} : \Delta_i. \left(\begin{array}{l}\Pi y_1 : \Phi_{i1}[\vec{x}] \to \mu(\vec{\Delta}).[\vec{\Phi}]. \\ \Pi h_{i1} : (\Pi\vec{z} : \Phi_{i1}[\vec{x}].C[y_1(\vec{z})]). \\ \cdots \\ \Pi y_{n_i} : \Phi_{in_i}[\vec{x}] \to \mu(\vec{\Delta}).[\vec{\Phi}]. \\ \Pi h_{in_i} : (\Pi\vec{z} : \Phi_{in_i}[\vec{x}].C[y_{n_i}(\vec{z})]).\end{array}\right) C[\mathsf{intro}_i^\mu(\vec{x}, y_1, \ldots, y_{n_i})] \\ for\ i = 1 \cdot\cdot n\end{array}}{\Gamma \vdash \mathsf{rec}_\mu(d_1, \ldots, d_n) \in \Pi z : \mu(\vec{\Delta}).[\vec{\Phi}].C}$$

Computation

$$\frac{\begin{array}{l}\Gamma, z : \mu(\vec{\Delta}).[\vec{\Phi}] \vdash C : Type \\ \Gamma \vdash d_i \in \Pi\vec{x} : \Delta_i. \left(\begin{array}{l}\Pi y_1 : \Phi_{i1}[\vec{x}] \to \mu(\vec{\Delta}).[\vec{\Phi}]. \\ \Pi h_{i1} : (\Pi\vec{z} : \Phi_{i1}[\vec{x}].C[y_1(\vec{z})]). \\ \cdots \\ \Pi y_{n_i} : \Phi_{in_i}[\vec{x}] \to \mu(\vec{\Delta}).[\vec{\Phi}]. \\ \Pi h_{in_i} : (\Pi\vec{z} : \Phi_{in_i}[\vec{x}].C[y_{n_i}(\vec{z})]).\end{array}\right) C[\mathsf{intro}_i^\mu(\vec{x}, y_1, \ldots, y_{n_i})] \\ for\ i = 1 \cdot\cdot n \\ \Gamma \vdash \vec{a} : \Delta_i,\ \Gamma \vdash b_1 : \Phi_{i1}[\vec{a}] \to \mu(\vec{\Delta}).[\vec{\Phi}], \ldots, \Gamma \vdash b_{n_i} : \Phi_{in_i}[\vec{a}] \to \mu(\vec{\Delta}).[\vec{\Phi}]\end{array}}{\Gamma \vdash \mathsf{rec}_\mu(\vec{d})(\mathsf{intro}_i^\mu(\vec{a}, \vec{b})) = \left(\begin{array}{c}d_i(\vec{a}, b_1, \lambda\vec{z} : \Phi_{i1}[\vec{a}].\mathsf{rec}_\mu(\vec{d})(b_1(\vec{z})), \ldots, \\ b_{n_i}, \lambda\vec{z} : \Phi_{in_i}[\vec{a}].\mathsf{rec}_\mu(\vec{d})(b_{n_i}(\vec{z})))\end{array}\right) : C[\mathsf{intro}_i^\mu(\vec{a}, \vec{b})]}$$

The formation and introduction rules tell us that the generalized inductive types are two dimensional generalization of the well-known W-types. The elimination and computation rules descibe how functions on those inductive types are defined and how they are computed. In the latter two rules, we have linearized what are completely independent to each other.

[3]This is why the formulation in terms of contexts is not entirely satisfactory. We however refrain from giving a completely formal account.

2.2 A Quick Review of the ω-Set Model

This section reviews some of the basic facts about the ω-Set model, cf. [Luo90a, Luo91, Ore92]. *In the sequel we will overload the syntax for the type theory and its interpretation in* ω-Set.

Definition 2.1 *An ω-set $(X, \Vdash_X)$ consists of a set X, the underlying set, and a relation $\Vdash_X \subset \omega \times X$ such that for any $a \in X$, there exists some $n \in \omega$ satisfying $n \Vdash_X a$ (every element of X is witnessed by some recursive function). A morphism from $(X, \Vdash_X)$ to $(Y, \Vdash_Y)$ is a function $f : X \longrightarrow Y$ that is* tracked *(*realized*) by some recursive function n. By "n tracks f" we mean that if $p \Vdash_X a$ then $n \cdot p \Vdash_Y fa$, where $n \cdot p$ is the result of applying the n-th recursive function to the number p.*

If A is an ω-set, we write $|A|$ for its underlying set. The category ω-Set is locally cartesian closed with finite colimits. Figure 1 summarizes some standard constructions. Perhaps more importantly, ω-Set contains a remarkable internal category.

Definition 2.2 *A* per*, partial equivalence relation, is a transitive symmetric relation A on the set ω of natural numbers. We write mAn for $(m,n) \in A$. $Q(A)$ is the set of equivalence classes of A. $[n]_A$ is the equivalence class represented by n. $Dom(A) \stackrel{\text{def}}{=} \{n | nAn\}$. $n \in A$ will mean $n \in Dom(A)$. A map from a per A to another per B is the set $[l]_{A \to B}$ of all natural numbers such that $\forall m, n \in [l]_{A \to B}. \forall p, q \in Dom(A). pAq \Rightarrow (m \cdot p) B (n \cdot q)$.*

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 $A \to B$ 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 *lccc* with finite colimits. But what makes Per particularly useful is the fact that it is a small complete full subcategory of ω-Set. This property enables us to interpret the *Calculus of Constructions* in ω-Set: types are interrpeted as ω-sets and propositions are interpreted as pers.

Therefore to show that the enriched *Calculus of Constructions* can be modeled in ω-Set, we only have to explain how the generalized inductive types are interpreted in this category. In the ω-Set model, a context is interpreted as an ω-set, a judgement $\Gamma \vdash A : \mathit{Type}$ is modeled by a map from the underlying set of Γ to ω-Set. The empty context is interpreted by the terminal ω-set $(\{\star\}, \omega \times \{\star\})$. The context $\Gamma, x : A$ is interpreted by $\sigma(\Gamma, A)$, where $A : |\Gamma| \longrightarrow \omega\text{-Set}$ is the denotation of $\Gamma \vdash A : \mathit{Type}$. $\Gamma \vdash \Pi x{:}A.B : \mathit{Type}$ is modeled by $\pi_\Gamma(A, B) : |\Gamma| \longrightarrow \omega\text{-Set}$. And the denotation of $\Gamma \vdash [x_1 : A_1, \ldots, x_n : A_n]\ \mathit{Ctxt}$ is given by $\sigma_\Gamma(A_1, \sigma_{\sigma(\Gamma, A_1)}(A_2, \cdots, \sigma_{\sigma(\cdots\sigma(\Gamma, A_1), \cdots, A_{n-2})}(A_{n-1}, A_n) \cdots))$. A term $\Gamma \vdash a : A$ is interpreted by a morphism $a : \Gamma \longrightarrow \sigma(\Gamma, A)$ that satisfies the *first projection property*: $a; \pi_1 = Id_\Gamma$ where $\pi_1 : \sigma(\Gamma, A) \longrightarrow \Gamma$ is the first projection. For details, see *loc.cit.*.

Assume Γ is an ω-set and $A, A' : |\Gamma| \longrightarrow \omega$-SET are two maps. We can define the ω-sets and $\pi(\Gamma, A)$ $\sigma(\Gamma, A)$ and two maps $A+A', A\times A' : |\Gamma| \longrightarrow \omega$-SET. If $B : |\sigma(\Gamma, A)| \longrightarrow \omega$-SET map, we can define two more maps is another $\sigma_\Gamma(A, B), \pi_\Gamma(A, B) : |\Gamma| \longrightarrow \omega$-SET. Notice $f : |\Gamma| \rightarrow thatamap\omega$-SET is defined by prescribing $f(\alpha)$ for an arbitrary $\alpha \in |\Gamma|$.

ω-set	the underlying set	the realizability relation
$\sigma(\Gamma, A)$	$\{(\gamma, a) \mid \gamma \in \lvert\Gamma\rvert \wedge a \in \lvert A(\gamma)\rvert\}$	$\langle m, n\rangle \Vdash_{\sigma(\Gamma,A)} (\gamma, a)$ if and only if $m \Vdash_\Gamma \gamma \wedge n \Vdash_{A(\gamma)} a$
$\sigma_\Gamma(A, B)(\alpha)$	$\{(a, b) \mid a \in \lvert A(\alpha)\rvert \wedge b \in \lvert B(\alpha, a)\rvert\}$	$\langle m, n\rangle \Vdash_{\sigma_\Gamma(A,B)(\alpha)} (a, b)$ if and only if $m \Vdash_{A(\alpha)} a \wedge n \Vdash_{B(\alpha,a)} b$
$\pi(\Gamma, A)$	$\{f : \lvert\Gamma\rvert \rightarrow \bigcup_{\gamma\in\lvert\Gamma\rvert} \lvert A(\gamma)\rvert \mid$ $\forall\gamma \in \lvert\Gamma\rvert. f(\gamma) \in \lvert A(\gamma)\rvert$ $\wedge \exists n \in \omega. n \Vdash_{\pi(\Gamma,A)} f\}$	$n \Vdash_{\pi(\Gamma,A)} f$ iff $\forall\gamma \in \lvert\Gamma\rvert. \forall p \in \omega.$ $p \Vdash_\Gamma \gamma \Rightarrow n \cdot p \Vdash_{A(\gamma)} f(\gamma)$
$\pi_\Gamma(A, B)(\alpha)$	$\{f : \lvert A(\alpha)\rvert \rightarrow \bigcup_{a\in\lvert A(\alpha)\rvert} \lvert B(\alpha, a)\rvert \mid$ $\forall a \in \lvert A(\alpha)\rvert. f(a) \in \lvert B(\alpha, a)\rvert$ $\wedge \exists n \in \omega. n \Vdash_{\pi_\Gamma(A,B)(\alpha)} f\}$	$n \Vdash_{\pi_\Gamma(A,B)(\alpha)} f$ if and only if $\forall a \in \lvert A(\alpha)\rvert. \forall p \in \omega.$ $p \Vdash_{A(\alpha)} a \Rightarrow n \cdot p \Vdash_{B(\alpha,a)} f(a)$
$(A + A')(\alpha)$	$\{(0, a) \mid a \in \lvert A(\alpha)\rvert\} \cup$ $\{(1, b) \mid b \in \lvert A'(\alpha)\rvert\}$	$\langle i, n\rangle \Vdash_{(A+A')(\alpha)} (j, c)$ if and only if $\begin{cases} i = j = 0 \wedge n \Vdash_{\lvert A(\alpha)\rvert} c \\ i = j = 1 \wedge n \Vdash_{\lvert A'(\alpha)\rvert} c \end{cases}$
$(A \times A')(\alpha)$	$\{(a, a') \mid a \in \lvert A(\alpha)\rvert \wedge a' \in \lvert A'(\alpha)\rvert\}$	$\langle m, n\rangle \Vdash_{(A\times A')(\alpha)} (a, a')$ if and only if $m \Vdash_{A(\alpha)} a \wedge n \Vdash_{A'(\alpha)} a'$

Notice that $\sigma_\Gamma(A, B)(\alpha) = \sigma(A(\alpha), B(\alpha, \_))$ and $\pi_\Gamma(A, B)(\alpha) = \pi(A(\alpha), B(\alpha, \_))$.

Figure 1: Some constructions in ω-SET

2.3 The Model

Definition 2.3 *([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 $\mathcal{I}(R) \stackrel{\text{def}}{=} \bigcap\{A | A \text{ is } R\text{-closed}\}$, the smallest R-closed set. A rule set R is* deterministic *if $\frac{u_1}{v} \in R \wedge \frac{u_2}{v} \in R \Rightarrow u_1 = u_2$.*

Associated with each rule set R is the R-induction: suppose P is a property; if for every rule $\frac{u}{v} \in R$, $\forall x \in u.P(x)$ implies that $P(v)$, then P holds for every member of $\mathcal{I}(R)$. This is a natural generalization of the transfinite induction in set theory. The corresponding generalization of definition by recursion is also available—to define a function on $\mathcal{I}(R)$, we do it on the way the elements of $\mathcal{I}(R)$ are generated.

Usually there is a more familiar way of obtaining what we want. We start from the empty set and keep throwing new elements into it. The transfinite sequence is bounded upwards often for simple cardinality reason. Therefore the sequence closes at some ordinal and the set obtained at this stage is $\mathcal{I}(R)$. It is obviously R-closed; on the other hand, each element of the set must be in all R-closed sets.

Let's look at the formation rule first. By assumption, $\Delta_i : |\Gamma| \longrightarrow \omega\text{-}\textsc{Set}$ for $i \in [1 \cdot\cdot n]$. For each $\alpha \in |\Gamma|$, $\Phi_{ik}(\alpha, \_) : |\Delta_i(\alpha)| \longrightarrow \omega\text{-}\textsc{Set}$ for $k \in [1 \cdot\cdot n_i]$.[4] Let V_κ be a sufficiently large universe. Define R_α to be the following rule set on V_κ:

$$\bigcup_{i\in[1\cdot\cdot n]} \left\{ \frac{\bigcup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik})}{\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i} \rangle} \;\middle|\; \begin{array}{l} \delta_i \in |\Delta_i(\alpha)| \wedge \\ \phi_{i1} \in |\Phi_{i1}(\alpha, \delta_i)| \to V_\kappa \\ \wedge \cdots \wedge \\ \phi_{in_i} \in |\Phi_{in_i}(\alpha, \delta_i)| \to V_\kappa \end{array} \right\}$$

where $n_{\mathsf{intro}_i} \in \omega$ is a code for intro_i and $range(\phi_{ik})$ is the range of the function ϕ_{ik}. What are the requirements on κ? The following analysis is again taken from [Dyb91]:

- For $\kappa \in \omega$, V_κ is finite. So κ must be no less than ω.
- V_κ should be closed under pairing. Because V_β is the greatest element in $V_{\beta+1}$, $\langle V_\beta, V_\beta \rangle \notin V_{\beta+1}$. We conclude that κ must be a limit ordinal.
- $|\Phi_{ij}(\alpha, \delta_i)| \to V_\kappa \subset V_\kappa$ for all possible i and j. This suggests that κ is regular.

If $\kappa' > \kappa$, we obtain another rule set R' on $V_{\kappa'}$ by replacing V_κ by $V_{\kappa'}$ in the above definition.

Proposition 2.4 *$\mathcal{I}(R_\alpha) = \mathcal{I}(R')$.*

[4]Suppose $\Delta = [x_1 : A_1, \ldots, x_n : A_n]$. Then by assumption, $\Phi_{ik} : |\sigma(\cdots\sigma(\sigma(\Gamma, A_1), A_2), \ldots, A_n)| \longrightarrow \omega\text{-}\textsc{Set}$. $\Phi_{ik}(\alpha, \_)$ is derived from Φ_{ik}.

Proof: Clearly $R_\alpha \subset R'$. Suppose $\frac{u}{v} \in R'$ and $u \subset \mathcal{I}(R_\alpha)$. Then $u = \cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik})$ by definition. So $u = \cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik}) \subset \mathcal{I}(R_\alpha) \subset V_\kappa$. Therefore $\phi_{ik} \in |\Phi_{ik}(\alpha, \delta_i)| \to V_\kappa$ for $k \in [1 \cdot\cdot n_i]$, $\alpha \in |\Gamma|$ and $\delta_i \in |\Delta_i(\alpha)|$. It follows that $\frac{u}{v} \in R_\alpha$. Conclude that $v \in \mathcal{I}(R_\alpha)$. That is $\mathcal{I}(R_\alpha)$ is R'-closed. On the other hand, if B is R'-closed, then by R_α-induction, it is easy to show that $\mathcal{I}(R_\alpha) \subset B$. Therefore $\mathcal{I}(R_\alpha)$ is the least R'-closed set. ■

The next proposition collects some properties of R_α and $\mathcal{I}(R_\alpha)$, which illustrates part of the reason why $\mathcal{I}(R_\alpha)$ models the inductive type.

Proposition 2.5 1. *R_α is deterministic.* 2. *If $x \in \mathcal{I}(R_\alpha)$, then $x = \langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle$ for some $i \in [1\cdot\cdot n_i]$, $\delta_i \in |\Delta_i(\alpha)|$ and $\phi_{ik} \in |\Phi_{ik}(\alpha, \delta_i)| \to V_\kappa$ where $k \in [1\cdot\cdot n_i]$.* 3. *Suppose $\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle \in \mathcal{I}(R_\alpha)$. Then there exists a unique rule $\frac{u}{\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle} \in R_\alpha$. Besides $u = \cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik})$.* 4. *Suppose $\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle \in \mathcal{I}(R_\alpha)$. Then ϕ_{ik} ($k \in [1 \cdot\cdot n_i]$) is a function from $|\Phi_{ik}(\alpha, \delta_i)|$ to $\mathcal{I}(R_\alpha)$.*

Proof: 1. This is just the extensionality of functions. 2. The set of all conclusions of the rules in R_α is R_α-closed. 3. Immediate from 1, 2 and the definition of R_α. 4. Suppose $\cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik}) \not\subseteq \mathcal{I}(R_\alpha)$. There must exist some $s \in \cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik})$ that is not in $\mathcal{I}(R_\alpha)$. Consider $\mathcal{I}(R_\alpha) \setminus \{\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle\}$. For any $\frac{u}{v} \in R_\alpha$ such that $u \subset \mathcal{I}(R_\alpha) \setminus \{\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle\}$, $u \subset \mathcal{I}(R_\alpha)$. So $v \in \mathcal{I}(R_\alpha)$. But v cannot be $\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle$ for otherwise $u = \cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik})$ by 3, which is impossible because s is in $\cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik})$ but certainly not in u. It follows that $v \in \mathcal{I}(R_\alpha) \setminus \{\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle\}$ and the latter is R_α-closed. Contradiction. Hence $\cup_{k\in[1\cdot\cdot n_i]} range(\phi_{ik}) \subset \mathcal{I}(R_\alpha)$. ■

We are now going to construct from $\mathcal{I}(R_\alpha)$ an ω-set $\omega(\mathcal{I}(R_\alpha))$. The construction proceeds by first constructing a series of functions $r^\beta : \mathcal{I}(R_\alpha) \longrightarrow P(\omega)$, where β is an ordinal and $P(\omega)$ the power set of ω, and a series of ω-sets $\omega^0 \subset \omega^1 \subset \cdots$ derived from the functions.

1. The function r^0 associates an empty set to each element of $\mathcal{I}(R_\alpha)$. $\omega^0 \stackrel{\text{def}}{=} (\varnothing, \varnothing)$.

2. $r^{\beta+1}(\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle) \stackrel{\text{def}}{=} S_i$ where

$$S_i \stackrel{\text{def}}{=} \left\{ \langle n_{\mathsf{intro}_i}, n_{\delta_i}, n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}\rangle \;\middle|\; \begin{array}{l} \delta_i \in |\Delta_i(\alpha)| \wedge \\ \phi_{i1} \in |\Phi_{i1}(\alpha, \delta_i) \to \omega^\beta| \\ \wedge \cdots \wedge \\ \phi_{in_i} \in |\Phi_{in_i}(\alpha, \delta_i) \to \omega^\beta| \\ \wedge n_{\delta_i} \Vdash_{\Delta_i(\alpha)} \delta_i \\ \wedge n_{\phi_{i1}} \Vdash_{\Phi_{i1}(\alpha,\delta_i)\to\omega^\beta} \phi_{i1} \\ \wedge \cdots \wedge \\ n_{\phi_{in_i}} \Vdash_{\Phi_{in_i}(\alpha,\delta_i)\to\omega^\beta} \phi_{in_i} \end{array} \right\}.$$

$\omega^{\beta+1} \stackrel{\text{def}}{=} (|\omega^{\beta+1}|, \Vdash_{\omega^{\beta+1}})$ where $x \in |\omega^{\beta+1}|$ iff $r^{\beta+1}(x) \neq \varnothing$, $n \Vdash_{\omega^{\beta+1}} x$ iff $n \in r^{\beta+1}(x)$.

3. For a limit ordinal θ, let

$$r^\theta(x) \stackrel{\text{def}}{=} \textstyle\bigcup_{\beta<\theta} r^\beta(x) \quad \text{for all } x \in \mathcal{I}(R_\alpha)$$
$$\omega^\theta \stackrel{\text{def}}{=} (\textstyle\bigcup_{\beta<\theta} |\omega^\beta|, \bigcup_{\beta<\theta} \Vdash_{\omega^\beta}).$$

Given $s, t : \mathcal{I}(R_\alpha) \longrightarrow P(\omega)$, define $s \leq t$ iff $\forall x \in \mathcal{I}(R_\alpha).s(x) \subset t(x)$. Obviously, for ordinals $\beta' \in \beta''$, $r^{\beta'} \leq r^{\beta''}$. By simple cardinality argument, we know that there exists an ordinal β such that for all $\theta \geq \beta$, $r^\theta = r^\beta$. By the least ordinal theorem ([Lev79]), we can choose the least such β. Define $\omega(\mathcal{I}(R_\alpha))$ to be ω^β. We can now interpret $\mu(\Delta_1, \ldots, \Delta_n).[\Phi_{11}, \ldots, \Phi_{1n_1}; \ldots; \Phi_{n1}, \ldots, \Phi_{nn_n}]$ by $\lambda\alpha : |\Gamma|.\omega(\mathcal{I}(R_\alpha))$. The interpretation of the elimination rule will be accommodated by proposition 2.7.

The eliminators on $\mu(\vec{\Delta}).[\vec{\Phi}]$ should usually be interpreted on the way the elements of $|\omega(\mathcal{I}(R_\alpha))|$ are generated. This can be done because of

Proposition 2.6 *Let $\dot{R}_\alpha$ be obtained from R_α by removing all the rules whose conclusions are in $\mathcal{I}(R_\alpha) \setminus |\omega(\mathcal{I}(R_\alpha))|$. Then $|\omega(\mathcal{I}(R_\alpha))| = \mathcal{I}(\dot{R}_\alpha)$.*

Proof: 1. $|\omega(\mathcal{I}(R_\alpha))|$ is $\dot{R}_\alpha$-closed. Suppose $\frac{u}{v} \in \dot{R}_\alpha$ and $u \subset |\omega(\mathcal{I}(R_\alpha))|$. Clearly $v \in \mathcal{I}(R_\alpha)$. If $v \notin |\omega(\mathcal{I}(R_\alpha))|$, then $v \in \mathcal{I}(R_\alpha) \setminus |\omega(\mathcal{I}(R_\alpha))|$. But that would imply that $\frac{u}{v} \notin \dot{R}_\alpha$. 2. Suppose A is $\dot{R}_\alpha$-closed. By induction on the way the elements of $|\omega(\mathcal{I}(R_\alpha))|$ are constructed, one can easily show that $\omega^\beta \subset A$ for all ordinal β. ■

We have defined the ω-set that is the interpretation of the inductive type. We need also of course to interpret the elimination and computation rules. For that purpose, we define another rule set whose least closed set is the graph of the underlying function modeling the eliminator. Suppose $\alpha \in |\Gamma|$ and $\delta_i \in |\Delta_i(\alpha)|$. By assumption, $C : |\sigma(\Gamma, \lambda\alpha : |\Gamma|.\omega(\mathcal{I}(R_\alpha)))| \longrightarrow \omega$-SET and $\phi_{i1} : |\Phi_{i1}(\alpha, \delta_i)| \longrightarrow |\omega(\mathcal{I}(R_\alpha))|$. Therefore $C[(\alpha, \_)] : |\omega(\mathcal{I}(R_\alpha))| \longrightarrow \omega$-SET. Write $C[(\alpha, \phi_{i1}(\_)]$ for the composition of ϕ_{i1} and $C[(\alpha, \_)]$. F_α is defined as follows:

$$\bigcup_{i \in [1 \cdots n]} \left\{ \frac{\cup_{k \in [1 \cdots n_i]} \{\langle \phi_{ik}(x), h_{ik}(x)\rangle \mid x \in |\Phi_{ik}(\alpha, \delta_i)|\}}{\langle\langle n_{\text{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\alpha)(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle} \;\middle|\; \begin{array}{l} \delta_i \in |\Delta_i(\alpha)| \wedge \\ \phi_{i1} \in |\Phi_{i1}(\alpha, \delta_i) \to \omega(\mathcal{I}(R_\alpha))| \\ h_{i1} \in |\pi(\Phi_{i1}(\alpha, \delta_i), C[(\alpha, \phi_{i1}(\_))])| \\ \wedge \cdots \wedge \\ \phi_{in_i} \in |\Phi_{in_i}(\alpha, \delta_i) \to \omega(\mathcal{I}(R_\alpha))| \\ h_{in_i} \in |\pi(\Phi_{in_i}(\alpha, \delta_i), C[(\alpha, \phi_{in_i}(\_))])| \end{array} \right\}.$$

Proposition 2.7 *$\mathcal{I}(F_\alpha)$ is the graph of a function in $|\pi(\omega(\mathcal{I}(R_\alpha)), C[(\alpha, \_)])|$.*

Proof: By induction on the way the elements of $\omega(\mathcal{I}(R_\alpha))$ are generated, we will construct a collection of sets $G^0 \subset G^1 \subset \cdots \subset \mathcal{I}(F_\alpha)$. Suppose we have constructed the sets up to β such that they satisfy two conditions:

(i) G^θ ($\theta \leq \beta$) is the graph of some function whose domain is a subset ψ^θ of $|\omega(\mathcal{I}(R_\alpha))|$ and for each element x of ψ^θ, $\{G^\theta\}(x) \in |C[(\alpha, x)]|$.
(ii) If $\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\alpha)(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle \in G^{\theta+1}$, then for $k \in [1 \cdot\cdot n_i]$, the equation $h_{ik} = \lambda x : |\Phi_{ik}(\alpha, \delta_i)|.\{G^\theta\}(\phi_{ik}(x))$ holds.

Now suppose the following rule is in F_α, and $\cup_{k \in [1 \cdots n_i]} \{\langle \phi_{ik}(x), h_{ik}(x)\rangle \mid x \in |\Phi_{ik}(\alpha, \delta_i)|\} \subset G^\beta$.

$$\frac{\cup_{k \in [1 \cdots n_i]} \{\langle \phi_{ik}(x), h_{ik}(x)\rangle \mid x \in |\Phi_{ik}(\alpha, \delta_i)|\}}{\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\alpha)(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle}$$

1. By definition, $\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\alpha)(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle \in \mathcal{I}(F_\alpha)$. By the assumption (i), we know that for every $x \in |\Phi_{ik}(\alpha, \delta_i)|$, $h_{ik}(x) = \{G^\beta\}(\phi_{ik}(x))$ for $k \in [1 \cdot\cdot n_i]$. The extensionality enables us to conclude that $h_{ik} = \lambda x : |\Phi_{ik}(\alpha, \delta_i)|.\{G^\theta\}(\phi_{ik}(x))$ for $k \in [1 \cdot\cdot n_i]$.
2. By the assumption on G^β, 1 and the definition of d_i ($i \in [1 \cdot\cdot n]$), it is obvious that $d_i(\alpha)(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i}) \in |C[(\alpha, \langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle)]|$.
3. Suppose $\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\alpha)(\delta_i, \phi_{i1}, h'_{i1}, \ldots, \phi_{in_i}, h'_{in_i})\rangle \in G^\theta$, $\theta \leq \beta$, then by induction hypothesis,

$$h'_{ik} = \lambda x : |\Phi_{ik}(\alpha, \delta_i)|.\{G^\theta\}(\phi_{ik}(x)) = \lambda x : |\Phi_{ik}(\alpha, \delta_i)|.\{G^\beta\}(\phi_{ik}(x)) = h_{ik}$$

for every $k \in [1 \cdot\cdot n_i]$.
From 1, 2 and 3, we conclude that

$$G^{\beta+1} \stackrel{\text{def}}{=} \{\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\alpha)(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle\} \cup G^\beta$$

is a subset of $\mathcal{I}(F_\alpha)$ and satisfies the conditions (i) and (ii). The case for a limit ordinal is handled as usual. We therefore have a sequence

$$G^0 \subset G^1 \subset \cdots \subset G^\beta \subset G^{\beta+1} \subset \cdots.$$

The cardinality restriction implies that the sequence must closes at some stage, say, β. But G^β is clearly F_α-closed. So $G^\beta = \mathcal{I}(F_\alpha)$. That is $\mathcal{I}(F_\alpha)$ is the graph of some function.

To show that $\mathcal{I}(F_\alpha)$ is in $|\pi(\omega(\mathcal{I}(R_\alpha)), C[(\alpha, \_)])|$, we must show that it is tracked by some recursive function. Suppose $r \Vdash_\Gamma \alpha$. Define a function as follows:

$$
\begin{array}{l}
\textbf{case } (m)_0 \textbf{ of} \\
\quad n_{\mathsf{intro}_1} : (n_{d_1} \cdot r) \cdot ((m)_1, (m)_2, \Lambda l.f \cdot ((m)_2 \cdot l), \ldots, (m)_{n_1+1}, \Lambda l.f \cdot ((m)_{n_1+1} \cdot l)); \\
\quad \vdots \\
\quad n_{\mathsf{intro}_n} : (n_{d_n} \cdot r) \cdot ((m)_1, (m)_2, \Lambda l.f \cdot ((m)_2 \cdot l), \ldots, (m)_{n_n+1}, \Lambda l.f \cdot ((m)_{n_n+1} \cdot l)) \\
\textbf{end}.
\end{array}
$$

$\Lambda l.f \cdot ((m)_2 \cdot l)$ etc. are obtained by *s-m-n* theorem. Here are some observations: (i) By Church thesis, the function is effective on the variable f and m. (ii) It is effective to construct the above function from Γ, the reason being that given $\alpha \in |\Gamma|$ and $r \Vdash_\Gamma \alpha$, one can obtain $n_{d_1} \cdot r, \ldots, n_{d_n} \cdot r$ effectively from $n_{d_1}, \ldots, n_{d_n}$. (iii) As $(\omega, \cdot)$ is a partial combinatory algebra, there exists an effective procedure that computes an index e such that $e \cdot (f, m) \simeq$ the above function.[5] Using *s-m-n* theorem again, we can compute a Gödel index $\Lambda f \Lambda m.e \cdot (f, m)$ for $\lambda f \lambda m.e \cdot (f, m)$ from e. The recursion theorem then says that there is an (index of an) effective function $\mathbf{R}_\alpha$ such that the following holds:

$$\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f, m)) \simeq \Lambda m.e \cdot (\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f, m)), m) \tag{1}$$

Notice that by (i), (ii), (iii) and the effectiveness of the recursion theorem, $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f, m))$ is effectively computed. This point is important for the interpretation of the eliminators.

[5] $x \simeq y$ iff $(x \downarrow \Leftrightarrow y \downarrow) \wedge (x \downarrow \Rightarrow x = y)$.

There remains one important thing to be proved—for any $c \in |\omega(\mathcal{I}(R_\alpha))|$ and any $n \Vdash_{\omega(\mathcal{I}(R_\alpha))} c$, $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot n$ is defined and

$$\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot n \Vdash_{C[\alpha,c]} \mathcal{I}(F_\alpha)(c). \tag{2}$$

This is proved by induction on the way the elements of $\omega(\mathcal{I}(R_\alpha))$ are generated. An element of $|\omega(\mathcal{I}(R_\alpha))|$ must be of the form $\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle$ and any of its realizers must be, by definition, of the form $\langle n_{\mathsf{intro}_i}, n_{\delta_i}, n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}\rangle$. So we have to show that

$$\begin{aligned}&\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot \langle n_{\mathsf{intro}_i}, n_{\delta_i}, n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}\rangle \\ &\Vdash_{C[(\alpha,\langle n_{\mathrm{intro}_i},\delta_i,\phi_{i1},\ldots,\phi_{in_i}\rangle)]} \mathcal{I}(F_\alpha)(\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle).\end{aligned} \tag{3}$$

Suppose it is added to the definition of $\omega(\mathcal{I}(R_\alpha))$ at the stage $\beta+1$. Then by (1)

$$\begin{aligned}&\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot \langle n_{\mathsf{intro}_i}, n_{\delta_i}, n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}\rangle \\ \simeq\ &(\Lambda m.e \cdot (\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)), m)) \cdot \langle n_{\mathsf{intro}_i}, n_{\delta_i}, n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}\rangle \\ \simeq\ &(n_{d_i} \cdot r) \cdot (n_{\delta_i}, n_{\phi_{i1}}, \Lambda l.\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{i1}} \cdot l), \ldots, \\ &n_{\phi_{in_i}}, \Lambda l.\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{in_i}} \cdot l)).\end{aligned}$$

We want to show that the expression at the third line of the above partial equation is defined. We know that $\phi_{i1} \in |\Phi_{i1}(\alpha,\delta_i) \to \omega^\beta|$ implies $\phi_{i1} \in |\Phi_{i1}(\alpha,\delta_i) \to \omega(\mathcal{I}(R_\alpha))|$. So by the structural definition of the interpretation on the elimination (and computation) rule(s), $n_{d_i} \cdot r$ is defined on n_{δ_i}, $n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}$. Now if $l \Vdash_{\Phi_{i1}(\alpha,\delta_i)} \phi$, then $n_{\phi_{i1}} \cdot l \Vdash_{\omega^\beta} \phi_{i1}(\phi)$, which implies $n_{\phi_{i1}} \cdot l \Vdash_{\omega(\mathcal{I}(R_\alpha))} \phi_{i1}(\phi)$. This is added to the definition of $\omega(\mathcal{I}(R_\alpha))$ in a stage previous to $\beta+1$. So by induction hypothesis $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{i1}} \cdot l)$ is defined and $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{i1}} \cdot l) \Vdash_{C[\alpha,\phi_{i1}(\phi)]} \mathcal{I}(F_\alpha)(\phi_{i1}(\phi))$. Conclude $\Lambda l.\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m))(n_{\phi_{i1}} \cdot l) \Vdash_{\pi(\Phi_{i1}(\alpha,\delta_i), C[(\alpha,\phi_{i1}(\_))])} \lambda\phi.\mathcal{I}(F_\alpha)(\phi_{i1}(\phi))$. Therefore $n_{d_i} \cdot r$ is defined at $\Lambda l.\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{i1}} \cdot l)$. It follows that $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m))$ is defined at $\langle n_{\mathsf{intro}_i}, n_{\delta_i}, n_{\phi_{i1}}, \ldots, n_{\phi_{in_i}}\rangle$. By assumption, the definitions of $\omega^{\beta+1}$ and $\mathcal{I}(F_\alpha)$ and the property (ii) that $\mathcal{I}(F_\alpha)$ satisfies, we have

$$\begin{aligned}&(n_{d_i} \cdot r) \cdot (n_{\delta_i}, n_{\phi_{i1}}, \Lambda l.\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{i1}} \cdot l), \ldots, \\ &n_{\phi_{in_i}}, \Lambda l.\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m)) \cdot (n_{\phi_{in_i}} \cdot l)) \\ &\Vdash_{C[\alpha,\langle n_{\mathrm{intro}_i},\delta_i,\phi_{i1},\ldots,\phi_{in_i}\rangle]} \\ &d_i(\alpha)(\delta_i, \phi_{i1}, \lambda\phi.\mathcal{I}(F_\alpha)(\phi_{i1}(\phi)), \ldots, \phi_{in_i}, \lambda\phi.\mathcal{I}(F_\alpha)(\phi_{in_i}(\phi))) \\ &= \mathcal{I}(F_\alpha)(\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle).\end{aligned}$$

We conclude that (3) holds. That is $\mathcal{I}(F_\alpha)$ is tracked by $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m))$. ■

Theorem 2.8 *The above construction gives a sound interpretation to inductive types in* Constructions.

Proof: The generalized inductive type $\mu(\vec{\Delta}).[\vec{\Phi}]$ is interpreted as $\lambda\alpha : |\Gamma|.\omega(\mathcal{I}(R_\alpha)) : |\Gamma| \longrightarrow \omega\text{-}\textsc{Set}$. The eliminator is taken care of by the effective function $\lambda\alpha.(\alpha, \mathcal{I}(F_\alpha)) : \Gamma \longrightarrow \sigma(\Gamma, \pi_\Gamma(\lambda\alpha : |\Gamma|.\omega(\mathcal{I}(R_\alpha)), C))$ and the equation rule is sound by the very definition of $\mathbf{R}_\alpha(\Lambda f \Lambda m.e \cdot (f,m))$ and the equation (1). ■

Example 2.9 *Some of the basic sets in Martin-Löf's set theory can be defined as follows:*

$$\begin{aligned}
Unit &\stackrel{\text{def}}{=} \mu([]).[] \\
A+B &\stackrel{\text{def}}{=} \mu([x:A],[y:B]).[] \\
N &\stackrel{\text{def}}{=} \mu([],[]).[;[]] \\
\mathsf{List}(A) &\stackrel{\text{def}}{=} \mu([],[x:A]).[[];[]] \\
\mathsf{Ord} &\stackrel{\text{def}}{=} \mu([],[],[O:Type]).[;[];[O:Type]] \\
\Sigma x:A.B &\stackrel{\text{def}}{=} \mu([x:A,y:B]).[] \\
Wx:A.B &\stackrel{\text{def}}{=} \mu([x:A]).[[y:B]].
\end{aligned}$$

Notice the difference between an empty context, which is inhabited by (), and no context at all. If we apply the previous constructions to $\mu([],[]).[;[]]$*, what we get is as follows:* $[]$ *is interpreted as* $(\{\star\}, \omega\times\{\star\})$*. Suppose we code the* intro_0 *and* intro_1 *by* 0 *and* 1 *respectively and simplify a function* $\{\star\}\longrightarrow V$ *to an element of* V*. Then the rule set is*

$$R_N \stackrel{\text{def}}{=} \left\{\frac{}{\langle 0,\star\rangle}\right\} \cup \left\{ \frac{\{\langle \overbrace{1,\ldots,1}^{i}, \langle 0,\star\rangle \cdots\rangle\}}{\langle \underbrace{1,\ldots,1}_{i+1}, \langle 0,\star\rangle\cdots\rangle} \;\middle|\; i\in\omega \right\}.$$

The smallest R_N*-closed set is* $N \stackrel{\text{def}}{=} \{\langle \overbrace{1,\ldots,1}^{i}, \langle 0,\star\rangle\cdots\rangle | i\in\omega\}$*. We have to effectivize the set obtained.* N_0 *is the initial* ω*-set.* N_1 *is the* ω*-set* $(\{\langle 0,\star\rangle\}, \omega\times\{\langle 0,\star\rangle\})$*.* N_2 *is the* ω*-set* $(\{\langle 0,\star\rangle, \langle 1,\langle 0,\star\rangle\rangle\}, \omega\times\{\langle 0,\star\rangle,\langle 1,\langle 0,\star\rangle\rangle\})$ *etc.. Conclude that* $\omega(N)$ *is* $(N,\omega\times N)$*. So according to our interpretation, the denotation of* $\mu([],[]).[;[]]$ *is (isomorphic to) the standard natural numbers object in* ω-Set*.*

Similarly the interpretations of the first, second and sixth of the above defined types are isomorphic to those given by the well-established interpretation.

2.4 Inductive Families

In [Dyb91], a class of types—*inductive families*—is introduced, which bear some resemblance to the combination of the two variants of *tree* types proposed in [PS89]. On occasions, it is useful to be able to deal with a family of inductive types indexed over another type. We take the attitude that an indexing type is different from a type in a context. The former plays a special role.

Formation

$$\frac{\begin{array}{l}\Gamma\vdash Ind\ Ctxt \\ \Gamma\vdash\Delta_1\ Ctxt,\ \ \Gamma\Delta_1\vdash\Phi_{11}\ Ctxt,\ldots,\ \Gamma\Delta_1\vdash\Phi_{1n_1}\ Ctxt,\ldots, \\ \Gamma\vdash\Delta_n\ Ctxt,\ \ \Gamma\Delta_n\vdash\Phi_{n1}\ Ctxt,\ldots,\Gamma\Delta_n\vdash\Phi_{nn_n}\ Ctxt\end{array}}{\Gamma, ind:Ind\vdash\mu(\Delta_1,\ldots,\Delta_n).[\Phi_{11},\ldots,\Phi_{1n_1};\ldots;\Phi_{n1},\ldots,\Phi_{nn_n}]:Type}$$

We will often abbreviate $\mu(\Delta_1, \ldots, \Delta_n).[\Phi_{11}, \ldots, \Phi_{1n_1}; \ldots; \Phi_{n1}, \ldots, \Phi_{nn_n}]$ to $\mu(\vec{\Delta}).[\vec{\Phi}]$ or even to μ.

Introduction

$$\frac{\begin{array}{l}\Gamma\Delta_i \vdash p_i : Ind, \quad \Gamma\Delta_i\Phi_{ik} \vdash q_{ik} : Ind \quad k \in [1 \cdot\cdot n_i] \\ \Gamma \vdash \vec{a} : \Delta_i \quad \Gamma \vdash b_k \in \Pi\vec{z} : \Phi_{ik}[\vec{a}].\mu(\vec{\Delta}).[\vec{\Phi}][q_{ik}[\vec{a}, \vec{z}]], \;\; k \in [1 \cdot\cdot n_i]\end{array}}{\Gamma \vdash \mathsf{intro}^{\mu}_i(\vec{a}, b_1, \ldots, b_{n_i}) : \mu(\vec{\Delta}).[\vec{\Phi}][p_i[\vec{a}]]}(i = 1 \cdot\cdot n)$$

Elimination

$$\frac{\begin{array}{l}\Gamma, ind : Ind, z : \mu(\vec{\Delta}).[\vec{\Phi}] \vdash C : Type \\ \Gamma\Delta_i \vdash p_i : Ind, \quad \Gamma\Delta_i\Phi_{ik} \vdash q_{ik} : Ind \quad k \in [1 \cdot\cdot n_i] \\ \Gamma \vdash d_i \in \Pi\vec{x} : \Delta_i. \left(\begin{array}{l}\Pi y_1 : (\Pi\vec{z} : \Phi_{i1}[\vec{x}].\mu(\vec{\Delta}).[\vec{\Phi}][q_{i1}[\vec{x}, \vec{z}]]). \\ \Pi h_{i1} : (\Pi\vec{z} : \Phi_{i1}[\vec{x}].C[q_{i1}[\vec{x}, \vec{z}], y_1(\vec{z})]) \\ \cdots \\ \Pi y_{n_i} : (\Pi\vec{z} : \Phi_{in_i}[\vec{x}].\mu(\vec{\Delta}).[\vec{\Phi}][q_{in_i}[\vec{x}, \vec{z}]]). \\ \Pi h_{in_i} : (\Pi\vec{z} : \Phi_{in_i}[\vec{x}].C[q_{in_i}[\vec{x}, \vec{z}], y_{n_i}(\vec{z})]).\end{array}\right) C[p_i[\vec{x}], \mathsf{intro}^{\mu}_i(\vec{x}, \vec{y})] \\ for\ i = 1 \cdot\cdot n\end{array}}{\Gamma, ind : Ind \vdash \mathsf{rec}_\mu(d_1, \ldots, d_n) \in \Pi z : \mu(\vec{\Delta}).[\vec{\Phi}].C[ind, \vec{z}]}$$

Computation

$$\frac{\begin{array}{l}\Gamma, ind : Ind, z : \mu(\vec{\Delta}).[\vec{\Phi}] \vdash C : Type \\ \Gamma\Delta_i \vdash p_i : Ind, \quad \Gamma\Delta_i\Phi_{ik} \vdash q_{ik} : Ind \quad k \in [1 \cdot\cdot n_i] \\ \Gamma \vdash d_i \in \Pi\vec{x} : \Delta_i. \left(\begin{array}{l}\Pi y_1 : (\Pi\vec{z} : \Phi_{i1}[\vec{x}].\mu(\vec{\Delta}).[\vec{\Phi}][q_{i1}[\vec{x}, \vec{z}]]). \\ \Pi h_{i1} : (\Pi\vec{z} : \Phi_{i1}[\vec{x}].C[q_{i1}[\vec{x}, \vec{z}], y_1(\vec{z})]) \\ \cdots \\ \Pi y_{n_i} : (\Pi\vec{z} : \Phi_{in_i}[\vec{x}].\mu(\vec{\Delta}).[\vec{\Phi}][q_{in_i}[\vec{x}, \vec{z}]]). \\ \Pi h_{in_i} : (\Pi\vec{z} : \Phi_{in_i}[\vec{x}].C[q_{in_i}[\vec{x}, \vec{z}], y_{n_i}(\vec{z})]).\end{array}\right) C[p_i[\vec{x}], \mathsf{intro}^{\mu}_i(\vec{x}, \vec{y})] \\ for\ i = 1 \cdot\cdot n \\ \Gamma \vdash \vec{a} : \Delta_i, \quad \Gamma \vdash b_k \in \Pi\vec{z} : \Phi_{ik}[\vec{a}].\mu(\vec{\Delta}).[\vec{\Phi}][q_{ik}[\vec{a}, \vec{z}]], \;\; k \in [1 \cdot\cdot n_i]\end{array}}{\Gamma \vdash \mathsf{rec}_\mu(\vec{d})[p_i[\vec{a}]](\mathsf{intro}^{\mu}_i(\vec{a}, \vec{d})) = \left(\begin{array}{c}d_i(\vec{a}, b_1, \lambda\vec{z} : \Phi_{i1}[\vec{a}].\mathsf{rec}_\mu(\vec{d})[q_{i1}[\vec{a}, \vec{z}]](b_1(\vec{z})), \ldots \\ b_{n_i}, \lambda\vec{z} : \Phi_{in_i}[\vec{a}].\mathsf{rec}_\mu(\vec{d})[q_{in_i}[\vec{a}, \vec{z}]](b_{n_i}(\vec{z})))\end{array}\right) \in C[p_i[\vec{a}], \mathsf{intro}^{\mu}_i(\vec{a}, \vec{b})]}$$

for $i \in [1 \cdot\cdot n]$.

The formation rule defines a family of types indexed over Ind. The introduction rule says that the family of types are mutually dependent. The elimination and the computation rules look stupid because we have linearized things that are independent of each other. This collection of types are not easy to use—one has to define q's and p's coherently in order to get a nontrivial family of inductive types. The introduction rule makes it clear that $\mu[ind]$ is not inhabited if ind is not equal to $p_i[\vec{a}]$ for some $\vec{a} : \Delta_i$. If this happens, the elimination rule is not fully specified since $\mathsf{rec}_\mu(\vec{d})[ind]$ is not defined. In the sequel, we assume that every object of Ind is equal to some $p_i[\vec{a}]$.

2.5 Semantical Justification

The ω-Set model can be constructed similarly as for the generalized inductive types. Fix $\alpha \in |\Gamma|$. For the interpretation of $\mu(\vec{\Delta}).[\vec{\Phi}]$, we define the rule set R_α as follows, where

V_κ is a large enough universe:

$$\bigcup_{i\in[1\cdot\cdot n]}\left\{\frac{\cup_{k\in[1\cdot\cdot n_i]}\{\langle q_{ik}(\alpha,\delta_i,x),\phi_{ik}(x)\rangle \mid x\in|\Phi_{ik}(\alpha,\delta_i)|\}}{\langle p_i(\alpha,\delta_i),\langle n_{\mathsf{intro}_i},\delta_i,\phi_{i1},\ldots,\phi_{in_i}\rangle\rangle}\;\middle|\;\begin{array}{l}\delta_i\in|\Delta_i(\alpha)|\wedge\\ \phi_{i1}\in|\Phi_{i1}(\alpha,\delta_i)|\to V_\kappa\\ \wedge\cdots\wedge\\ \phi_{in_i}\in|\Phi_{in_i}(\alpha,\delta_i)|\to V_\kappa\end{array}\right\}.$$

Let $\langle ind',x'\rangle$ and $\langle ind'',x''\rangle$ be two elements in $\mathcal{I}(R_\alpha)$. Then $ind'\neq ind''$ implies $x'\neq x''$. For suppose $ind'=p_i(\alpha,\delta_i')$ and $ind''=p_j(\alpha,\delta_j'')$, if $i\neq j$ then clearly $x'\neq x''$; otherwise we must have $\delta_i'\neq\delta_i''$ which also implies $x'\neq x''$. So $\mathcal{I}(R_\alpha)$ defines a function.

As before we take the 'effective part' of the set $\mathcal{I}(R_\alpha)$ by a transfinite construction:

1. $\omega^0_{ind}\stackrel{\mathrm{def}}{=}(\varnothing,\varnothing)$ for all $ind\in|Ind(\alpha)|$.
2. Suppose ω^β_{ind} has been constructed for every $ind\in|Ind(\alpha)|$. Let $ind\stackrel{\mathrm{def}}{=}p_i(\alpha,\delta_i)$. Define $\omega^{\beta+1}_{ind}\stackrel{\mathrm{def}}{=}\omega^\beta_{ind}\cup S^\beta_{ind}$ where S^β_{ind} is

$$\bigcup_{i\in[1\cdot\cdot n]}\left\{\begin{array}{l}\langle\langle n_{\mathsf{intro}_i},n_{\delta_i},n_{\phi_{i1}},\ldots,n_{\phi_{in_i}}\rangle,\\ \;\langle n_{\mathsf{intro}_i},\delta_i,\phi_{i1},\ldots,\phi_{in_i}\rangle\rangle\end{array}\;\middle|\;\begin{array}{l}\delta_i\in|\Delta_i(\alpha)|\wedge\\ \phi_{i1}\in|\pi(\Phi_{i1}(\alpha,\delta_i),\omega^\beta_{q(\alpha,\delta_i,\_)})|\\ \wedge\cdots\wedge\\ \phi_{in_i}\in|\pi(\Phi_{in_i}(\alpha,\delta_i),\omega^\beta_{q(\alpha,\delta_i,\_)})|\\ \wedge n_{\delta_i}\Vdash_{\Delta_i(\alpha)}\delta_i\\ \wedge n_{\phi_{i1}}\Vdash_{\pi(\Phi_{i1}(\alpha,\delta_i),\omega^\beta_{q(\alpha,\delta_i,\_)})}\phi_{i1}\\ \wedge\cdots\wedge\\ n_{\phi_{in_i}}\Vdash_{\pi(\Phi_{in_i}(\alpha,\delta_i),\omega^\beta_{q(\alpha,\delta_i,\_)})}\phi_{in_i}\end{array}\right\}.$$

3. For a limit ordinal θ, $\omega^\theta_{ind}\stackrel{\mathrm{def}}{=}(\bigcup_{\beta<\theta}|\omega^\beta_{ind}|,\bigcup_{\beta<\theta}\Vdash_{\omega^\beta_{ind}})$ for each $ind\in|Ind(\alpha)|$.

We take the least fixpoint $\{\omega(\mathcal{I}(R_\alpha)(ind))\}_{ind\in|Ind(\alpha)|}$ of the sequence. $\mu(\vec{\Delta}).[\vec{\Phi}]$ is interpreted as the function that sends $(\alpha,ind)\in|\sigma(\Gamma,Ind)|$ to $\omega(\mathcal{I}(R_\alpha)(ind))$. The terms of $\mu(\vec{\Delta}).[\vec{\Phi}]$ are interpreted using the closure property of $\omega(\mathcal{I}(R_\alpha)(ind))$.

The ω-Set for the graph of the interpretation and the Gödel number that tracks it can be constructed in the same fashion. First we construct the rule set F_α:

$$\bigcup_{i\in[1\cdot\cdot n]}\left\{\begin{array}{l}\dfrac{\cup_{k\in[1\cdot\cdot n_i]}\{\langle q_{ik}(\alpha,\delta_i,x),\langle\phi_{ik}(x),h_{ik}(x)\rangle\rangle|x\in|\Phi_{ik}(\alpha,\delta_i)|\}}{\langle p_i(\alpha,\delta_i),\langle\langle n_{\mathsf{intro}_i},\delta_i,\phi_{i1},\ldots,\phi_{in_i}\rangle,d_i(\delta_i,\phi_{i1},h_{i1},\ldots,\phi_{in_i},h_{in_i})\rangle\rangle}\\ \left|\begin{array}{l}\delta_i\in|\Delta_i(\alpha)|\wedge\\ \phi_{i1}\in|\pi(\Phi_{i1}(\alpha,\delta_i),\omega(\mathcal{I}(R_\alpha)(q_{i1}(\alpha,\delta_i,\_))))|\wedge\\ h_{i1}\in|\pi(\Phi_{i1}(\alpha,\delta_i),C(\alpha,q_{i1}(\alpha,\delta_i,\_),\phi_{i1}(\_)))|\\ \wedge\cdots\wedge\\ \phi_{in_i}\in|\pi(\Phi_{in_i}(\alpha,\delta_i),\omega(\mathcal{I}(R_\alpha)(q_{in_i}(\alpha,\delta_i,\_))))|\wedge\\ h_{in_i}\in|\pi(\Phi_{in_i}(\alpha,\delta_i),C(\alpha,q_{in_i}(\alpha,\delta_i,\_),\phi_{in_i}(\_)))|\end{array}\right.\end{array}\right\}.$$

$\mathcal{I}(F_\alpha)$ can also be obtained from the following transfinite sequences

$$G^0_{ind}\subset G^1_{ind}\subset\cdots\subset G^\alpha_{ind}\tag{4}$$

for all $ind\in|Ind(\alpha)|$. (4) is defined as follows:

- G^0_{ind} is the empty set.
- Suppose G^θ_{ind} has been constructed for $\theta \le \beta$ and $ind \in |Ind(\alpha)|$. Then for $\delta_i \in |\Delta(\alpha)|$, $G^{\beta+1}_{p_i(\alpha,\delta_i)}$ is defined as the union of $G^\beta_{p_i(\alpha,\delta_i)}$ and

$$\left\{ \begin{array}{l|l} & \phi_{i1} \in |\pi(\Phi_{i1}(\alpha,\delta_i), \omega^\beta_{q(\alpha,\delta_i,\_)})| \\ \langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, & h_{i1} \in |\pi(\Phi_{i1}(\alpha,\delta_i), C(\alpha, q_{i1}(\alpha,\delta_i,\_), \phi_{i1}(\_)))| \\ d_i(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle & \wedge \cdots \wedge \\ & \phi_{in_i} \in |\pi(\Phi_{in_i}(\alpha,\delta_i), \omega^\beta_{q(\alpha,\delta_i,\_)})| \\ & h_{in_i} \in |\pi(\Phi_{in_i}(\alpha,\delta_i), C(\alpha, q_{in_i}(\alpha,\delta_i,\_), \phi_{in_i}(\_)))| \end{array} \right\}.$$

- For a limit ordinal γ,

$$G^\gamma_{ind} \stackrel{\text{def}}{=} \cup_{\theta\in\gamma} G^\theta_{ind}.$$

Proposition 2.10 *1. G^β_{ind} is the graph of a function on $|\omega^\beta_{ind}|$. 2. If $\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle \in G^{\beta+1}_{p_i(\alpha,\delta_i)}$ then $h_{ik} = \lambda x : |\Phi_{ik}(\alpha,\delta_i)|.\{G^\beta_{q_{ik}(\alpha,\delta_i,x)}\}(\phi_{ik}(x))$. 3. The closure ordinal for (4) is the same as that for $\omega(\mathcal{I}(R_\alpha)(ind))$.*

Proof: We will prove *1* and *2* simultaneously by induction. Suppose they hold for $\theta \le \beta$. If

$$\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle \in G^{\beta+1}_{p_i(\alpha,\delta_i)} \setminus G^\beta_{p_i(\alpha,\delta_i)},$$

then by induction hypothesis on *1*,

$$\begin{aligned} h_{ik} &= \lambda x : |\Phi_{ik}(\alpha,\delta_i)|.h_{ik}(x) \\ &= \lambda x : |\Phi_{ik}(\alpha,\delta_i)|.\{G^\beta_{g_{ik}(\alpha,\delta_i,x)}\}(\phi_{ik}(x)). \end{aligned}$$

$G^{\beta+1}_{p_i(\alpha,\delta_i)}$ is a graph on $|\omega^{\beta+1}_{p_i(\alpha,\delta_i)}|$ as for $\langle\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle, d_i(\delta_i, \phi_{i1}, h_{i1}, \ldots, \phi_{in_i}, h_{in_i})\rangle \in G^{\beta+1}_{p_i(\alpha,\delta_i)} \setminus G^\beta_{p_i(\alpha,\delta_i)}$, $\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle$ is not in $|\omega^\beta_{p_i(\alpha,\delta_i)}|$.

3 is immediate from *1*.

We omit the the argument, which is similar to the one given in section 2.3, that this function is tracked by some effective function. ■

2.6 The Recursive Model

The method described in section 2.3 can also be applied to PER. What we get is a model of Martin-Löf's set theory extended with the generalized inductive types. To model universes, we must associate a name to each per that is the interpretation of some type in ML^∞, the Martin-Löf's set theory with an infinite hierarchy of universes. Then a type appeared on the left hand side of ':' is interpreted as a name; and the same type is modeled by the corresponding per when it appears to the right of ':'. This is basically the extension of the recursive model given in [Bee82, Bee85].

In the per-setting, two points are worth mentioning: (i) because of the double role a type plays in Martin-Löf's set theory with small universes, there is a tremendous amount of encoding involved; if we want to model the eliminators for the universes, the problem is even more severe; (ii) the universe V_κ in the definition of the rule sets can always be taken to be the terminal per $\omega\times\omega$, which is the largest element in the poset $(Obj(\textsc{Per}), \subseteq)$. So typically a rule set R_α is defined as follows:

$$\bigcup_{i\in[1\cdots n]} \left\{ \frac{\cup_{k\in[1\cdots n_i]} range(\phi_{ik})}{\langle n_{\mathsf{intro}_i}, \delta_i, \phi_{i1}, \ldots, \phi_{in_i}\rangle} \;\middle|\; \begin{array}{l} \delta_i \in \Delta_i(\alpha) \wedge \\ \phi_{i1} \in \Phi_{i1}(\alpha, \delta_i) \to \omega\times\omega \\ \wedge\cdots\wedge \\ \phi_{in_i} \in \Phi_{in_i}(\alpha, \delta_i) \to \omega\times\omega. \end{array} \right\}.$$

Notice that in the above definition all the elements and the tuples are natural numbers. A transfinite induction should be used to define a partial equivalence relation on $\mathcal{I}(R_\alpha)$. This process produces a per $\mathcal{P}(\mathcal{I}(R_\alpha))$. For details, see [Fu92c].

2.7 Models of Other Variants

Inductive types defined using a logical framework. One of the advantages of using a logical framework is the possibility of separating the computational content of a language to be defined from the extensional properties of the meta calculus. The language proposed in [Luo92] is designed to maximize this gain. To be able to define the generalized inductive types, the Martin-Löf's logical framework is extended with kind schemata. These schemata are 'small' in the sense that they only talk about the small universe *Set*. The generalized inductive types, which reside in *Set*, are then defined in terms of the kind schemata.

From a model theoretic point of view, *Set* is an internal category of the ambient category within which the logical framework is interpreted. So as usual we can interpret *Set* as the ω-Set and kinds as objects of the category Ω-SET of ω-sets defined on a large universe. The rule sets as defined in § 2.3 live in Ω-SET. The 'smallness' of the kind schemata implies that the least closed sets of these rule sets are in ω-Set. For more about categorical semantics of logical frameworks, see [Fu93a].

Inductive types via W-types. It is well-known that inductive types are definable in an extensional dependent calculus with W-types ([Dyb89, GL92]). The interpretation given in § 2.3 is extensional. It follows that it also interprets the W-types with the extensional rules. So the extensional *Calculus of Constructions* enriched with W-types is accounted for by the method illustrated in § 2.3.

Other formulations of generalized inductive types. Once the essence of the generalized inductive types is understood, different formulations should present no problems. [CPM90] and [PPM91] contain two other formulations. The advantage of the formulation in [CPM90] is that it is easier to see how to generalize(!) the generalized inductive types. We call *general inductive types* those inductive types defined in [CPM90] with strict positivity being relaxed to just positivity.

With little modification, our ω-Set model is a sound interpretation of the generalized inductive types formulated in [CPM90]. Our method, however, does *not* give a model of the *general* inductive types in ω-Set. But it *does* give a model of the *general* inductive types in Per. This is due to the existence of a largest universe, as it were, in Per. For more details, see [Fu92c].

Internal Inductive Types. A totally different approach is to code up inductive types internally. This is adopted in [Wra89, PPM91, Fu93b]. This method does not pose any model theoretical problems in the traditional sense. The semantic interest here is about how to relate two levels of models. See [Pit87, Fu93a, Fu92a] for details.

Categorical Inductive Types. In [Men91], a categorical formulation of recursion is given. It is interesting to see what this general definition means in concrete categories in which dependent typed calculus can be modeled. See [Fu92b] for some examples.

3 The Initial Algebraic Approach

There is another way of formalizing the ideas embodied in the generalized inductive types by using initial T-algebras. This is studied in [Ore92]. For this formulation to be possible, the calculus must be extensional. The reason is that we use functors to define rules about the inductive types and the functoriality of a type constructor is equivalent to the extensionality of the definitional equality associated with that type constructor. In this section we first spell out the rules for *mutually dependent inductive types*. It should be clear why we chose **ECC** to be the ambient calculus. Then two examples are given to illustrate the usage of these rules. Next we explain a general method to construct the interpretation of those types. We give a nice generalization of a well-known result. This extra generality then motivates the definition of a new class of inductive types to be described in the next section.

3.1 The Language

Definition 3.1 *Suppose $X : Type_i \vdash \phi(X) : Type_i$. 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 $X : Type_i \vdash \phi(X) : Type_i$ 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 $K_1, \cdots, K_m$.*

Formation

$$\frac{\Gamma, X_1 : Type_i, \ldots, X_n : Type_i \vdash M_1 : Type_i\ , \ldots,\ \Gamma, X_1 : Type_i, \ldots, X_n : Type_i \vdash M_n : Type_i}{\Gamma \vdash \mu^i \vec{X}.(M_1, \ldots, M_n)_j : Type_i}$$

where $X_1, \ldots, X_n$ appear strictly positively in $M_1, \ldots, M_n$.

Introduction

$$\frac{\begin{array}{l}\Gamma, X_1 : \mathit{Type}_i, \ldots, X_n : \mathit{Type}_i \vdash M_1 : \mathit{Type}_i \;, \ldots,\; \Gamma, X_1 : \mathit{Type}_i, \ldots, X_n : \mathit{Type}_i \vdash M_n : \mathit{Type}_i, \\ \Gamma \vdash N : M_j[\mu^i_1/X_1, \ldots, \mu^i_n/X_n]\end{array}}{\Gamma \vdash \mathsf{intro}^{\mu^i}_j(N) : \mu^i \vec{X}.(M_1, \ldots, M_n)_j}$$

where we have abbreviated $\mu^i \vec{X}.(M_1, \ldots, M_n)_j$ to μ^i_j.

Recursion

$$\frac{\begin{array}{l}\Gamma, x_1 : \mu^i \vec{X}.(\vec{M})_1 \vdash C_1 : \mathit{Type}_i \;, \ldots,\; \Gamma, x_n : \mu^i \vec{X}.(\vec{M})_n \vdash C_n : \mathit{Type}_i \\ \Gamma \vdash f_1 : \Pi z\!:\! M_1[\Sigma x_1 : \mu^i_1.C_1/X_1, \ldots, \Sigma x_n : \mu^i_n.C_n/X_n].C_1[\mathsf{intro}^{\mu^i}_1((M^\star_1(\vec{\pi_1}))z)/x_1] \\ \vdots \\ \Gamma \vdash f_n : \Pi z\!:\! M_n[\Sigma x_1 : \mu^i_1.C_1/X_1, \ldots, \Sigma x_n : \mu^i_n.C_n/X_n].C_n[\mathsf{intro}^{\mu^i}_n((M^\star_n(\vec{\pi_1}))z)/x_n]\end{array}}{\Gamma \vdash \mathsf{Rec}^{\mu^i}_j(\vec{f}) : \Pi x_j \!:\! \mu^i \vec{X}.(M_1, \ldots, M_n)_j.C_j}$$

where $\vec{\pi_1} \stackrel{\text{def}}{=} \lambda w_1 : (\Sigma x_1 : \mu^i_1.C_1).\pi_1 w_1, \ldots, \lambda w_n : (\Sigma x_n : \mu^i_n.C_n).\pi_1 w_n$. $M^\star_j$, $j \in [1, \ldots, n]$, is the n-ary functor given by M_j. It is covariant because M_j is strictly positive with respect to $X_1, \ldots, X_n$.

Computation

$$\frac{\begin{array}{l}\Gamma, x_1 : \mu^i \vec{X}.(\vec{M})_1 \vdash C_1 : \mathit{Type}_i \;, \ldots,\; \Gamma, x_n : \mu^i \vec{X}.(\vec{M})_n \vdash C_n : \mathit{Type}_i \\ \Gamma \vdash N : M_j[\mu^i_1/x_1, \ldots, \mu^i_n/x_n] \\ \Gamma \vdash f_1 : \Pi z\!:\! M_1[\Sigma x_1 : \mu^i_1.C_1/X_1, \ldots, \Sigma x_n : \mu^i_n.C_n/X_n].C_1[\mathsf{intro}^{\mu^i}_1((M^\star_1(\vec{\pi_1}))z)/x_1] \\ \vdots \\ \Gamma \vdash f_n : \Pi z\!:\! M_n[\Sigma x_1 : \mu^i_1.C_1/X_1, \ldots, \Sigma x_n : \mu^i_n.C_n/X_n].C_n[\mathsf{intro}^{\mu^i}_n((M^\star_n(\vec{\pi_1}))z)/x_n]\end{array}}{\Gamma \vdash \mathsf{Rec}_j(\vec{f})(\mathsf{intro}^{\mu^i}_j(N)) {=} f_j(M^\star_j(\lambda z{:}\mu^i_1.\langle z, \mathsf{Rec}^{\mu^i}_1(\vec{f})z\rangle(N), \ldots, \lambda z{:}\mu^i_n.\langle z, \mathsf{Rec}_n(\vec{f})z\rangle(N))) : C_j[\mathsf{intro}^{\mu^i}_j(N)/x_j]}$$

for $j \in [1, \ldots, n]$.

Extensionality

$$\frac{\begin{array}{l}\Gamma, x_1 : \mu^i \vec{X}.(\vec{M})_1 \vdash_\Delta C_1 : \mathit{Type}_i \;, \ldots,\; \Gamma, x_n : \mu^i \vec{X}.(\vec{M})_n \vdash_\Delta C_n : \mathit{Type}_i, \\ \Gamma \vdash \lambda z : \mu^i_1.g_1(\mathsf{intro}^{\mu^i}_1(z)) = \lambda z : \mu^i_1.f_1(M^\star_1(\langle \vec{id, g}\rangle)z) \in \Pi x_1 : \mu^i \vec{X}.(M_1, \ldots, M_n)_1.C_1 \\ \vdots \\ \Gamma \vdash \lambda z : \mu^i_n.g_n(\mathsf{intro}^{\mu^i}_n(z)) = \lambda z : \mu^i_n.f_n(M^\star_n(\langle \vec{id, g}\rangle)z) \in \Pi x_n : \mu^i \vec{X}.(M_1, \ldots, M_n)_n.C_n \\ \Gamma \vdash f_1 : \Pi z\!:\! M_1[\Sigma x_1 : \mu^i_1.C_1/X_1, \ldots, \Sigma x_n : \mu^i_n.C_n/X_n].C_1[\mathsf{intro}^{\mu^i}_1((M^\star_1(\vec{\pi_1}))z)/x_1] \\ \vdots \\ \Gamma \vdash f_n : \Pi z\!:\! M_n[\Sigma x_1 : \mu^i_1.C_1/X_1, \ldots, \Sigma x_n : \mu^i_n.C_n/X_n].C_n[\mathsf{intro}^{\mu^i}_n((M^\star_n(\vec{\pi_1}))z)/x_n]\end{array}}{\Gamma \vdash \mathsf{Rec}^{\mu^i}_j(\vec{f}) = g_j : \Pi x_j \!:\! \mu^i \vec{X}.(M_1, \ldots, M_n)_j.C_j}$$

where $\langle \vec{id, g}\rangle \stackrel{\text{def}}{=} \lambda z : \mu^i_n.\langle z, g_n z\rangle, \ldots, \lambda z : \mu^i_n.\langle z, g_n z\rangle$.

The following diagram helps to understand the rules concerning the inductive types.

$$\begin{tikzcd}
M_j[\mu_1,\ldots,\mu_n] \arrow[r, "M_j^\star(\langle id, \mathsf{Rec}_1^\mu(\vec{f})\rangle,\ldots,\langle id, \mathsf{Rec}_n^\mu(\vec{f})\rangle)"] \arrow[d, "\mathsf{intro}_j^\mu"'] & M_j[\Sigma x_1:\mu_1.C_1,\ldots,\Sigma x_n:\mu_n.C_n] \arrow[d, "\langle M_j^\star(\vec{\pi_1});\mathsf{intro}_j^\mu, f_j\rangle"] \\
\mu_j \arrow[r, "\langle id, \mathsf{Rec}_j^\mu(\vec{f})\rangle"'] & \Sigma x_j:\mu_j.C_j
\end{tikzcd}$$

Example 3.2 *The best known mutually dependent inductive type is the data type of trees and forests. Using Hagino's notation, it is defined as follows (intuitively, a tree consists of a forest and a root; there exists an empty forest; adjoining a tree to a forest creates a new forest):*

$$\begin{array}{rl} \langle T, F\rangle \equiv & \textbf{sum } (X,Y) \textbf{ with constructors} \\ & \quad \mathsf{span} : A\times Y \longrightarrow X; \\ & \quad \mathsf{join} : X\times Y \longrightarrow Y, \\ & \quad \mathsf{empty} : \mathbf{1} \longrightarrow Y \\ & \textbf{end}. \end{array}$$

In our notation, some of the rules are as follows:

$$\frac{X:Type_i, Y:Type_i \vdash A\times Y : Type_i \quad X:Type_i, Y:Type_i \vdash \mathbf{1}+X\times Y : Type_i}{\vdash \mathsf{Tree}(A) \stackrel{\text{def}}{=} \mu^i(X,Y).(A\times Y, \mathbf{1}+X\times Y)_1}$$

$$\frac{X:Type_i, Y:Type_i \vdash A\times Y : Type_i \quad X:Type_i, Y:Type_i \vdash \mathbf{1}+X\times Y : Type_i}{\vdash \mathsf{Forest}(A) \stackrel{\text{def}}{=} \mu^i(X,Y).(A\times Y, \mathbf{1}+X\times Y)_2}$$

$$\frac{X:Type_i, Y:Type_i \vdash A\times Y : Type_i \quad X:Type_i, Y:Type_i \vdash \mathbf{1}+X\times Y : Type_i \quad \vdash t : A\times \mathsf{Forest}(A)}{\vdash tree(t) : \mathsf{Tree}(A)}$$

$$\frac{X:Type_i, Y:Type_i \vdash A\times Y : Type_i \quad X:Type_i, Y:Type_i \vdash \mathbf{1}+X\times Y : Type_i \quad \vdash f : \mathbf{1}+\mathsf{Tree}(A)\times\mathsf{Forest}(A)}{\vdash forest(f) : \mathsf{Forest}(A)}$$

$$\frac{\begin{array}{l} \vdash f : \mathbf{1}+\mathsf{Tree}(A)\times\mathsf{Forest}(A) \\ x_T : \mathsf{Tree}(A) \vdash C_T : Type_i \quad x_F : \mathsf{Forest}(A) \vdash C_F : Type_i \\ \vdash f_T : \Pi z{:}A\times(\Sigma x_F : \mathsf{Forest}(A).C_F).C_T[tree(\langle \pi_1 z, \pi_1\pi_2 z\rangle)/x_T] \\ \vdash f_F : \Pi z{:}\mathbf{1}+(\Sigma x_T : \mathsf{Tree}(A).C_T)\times(\Sigma x_F : \mathsf{Forest}(A).C_F). \\ C_F[forest(\mathbf{case}(z, \lambda w:\mathbf{1}.\mathsf{inl}(w), \lambda w:(\Sigma x_T:\mathsf{Tree}(A).C_T)\times \\ (\Sigma x_F : \mathsf{Forest}(A).C_F).\mathsf{inr}(\langle \pi_1\pi_1 w, \pi_1\pi_2 w\rangle))))/x_F] \end{array}}{\vdash \mathsf{Rec}_F(f_T,f_F)(forest(f)) = \begin{array}{l} f_F(\mathbf{case}(f, \mathsf{Inl}(x).\mathsf{In}(\star), \mathsf{Inr}(x).\langle\langle \pi_1 x, \mathsf{Rec}_T(f_T,f_F)(\pi_1 x)\rangle, \\ \quad \langle \pi_2 x, \mathsf{Rec}_F(f_T,f_F)(\pi_2 x)\rangle\rangle)) : C_F[forest(f)/x_F] \end{array}}$$

$$\frac{\begin{array}{l} \vdash t : A\times\mathsf{Forest}(A) \\ x_T : \mathsf{Tree}(A) \vdash C_T : Type_i \quad x_F : \mathsf{Forest}(A) \vdash C_F : Type_i \\ \vdash f_T : \Pi z{:}A\times(\Sigma x_F : \mathsf{Forest}(A).C_F).C_T[tree(\langle \pi_1 z, \pi_1\pi_2 z\rangle)/x_T] \\ \vdash f_F : \Pi z{:}\mathbf{1}+(\Sigma x_T : \mathsf{Tree}(A).C_T)\times(\Sigma x_F : \mathsf{Forest}(A).C_F). \\ C_F[forest(\mathbf{case}(z, \lambda w:\mathbf{1}.\mathsf{inl}(w), \lambda w:(\Sigma x_T:\mathsf{Tree}(A).C_T)\times \\ (\Sigma x_F : \mathsf{Forest}(A).C_F).\mathsf{inr}(\langle \pi_1\pi_1 w, \pi_1\pi_2 w\rangle))))/x_F] \end{array}}{\vdash \mathsf{Rec}_T(f_T,f_F)(tree(t)) = f_T(\langle \pi_1 t, \langle \pi_2 t, \mathsf{Rec}_F(f_T,f_F)(\pi_2 t)\rangle\rangle) : C_T[tree(t)/x_T]}.$$

The rules are not as complicated as they appear to be.

Example 3.3 *(continued) Suppose A is the type of natural numbers N. We can compute the sum of all the nodes and leaves of a tree and the sum of all the nodes and leaves of all the trees in a forest. The two mutually recursive programmes are derived as follows:*

$$\frac{\begin{array}{l}\vdash t : N \times \mathsf{Forest}(N)\\ \vdash f_T \stackrel{\text{def}}{=} \lambda x : N \times (\mathsf{Forest}(N) \times N).\pi_1 x + \pi_2\pi_2 x : N \times (\mathsf{Forest}(N) \times N) \to N\\ \vdash f_F \stackrel{\text{def}}{=} \lambda x : \mathbf{1} + (\mathsf{Tree}(N) \times N) \times (\mathsf{Forest}(N) \times N).\\ \quad \mathbf{case}(x, \mathsf{Inl}(z).0, \mathsf{Inr}(z).\pi_2\pi_1 z + \pi_2\pi_2 z) : \mathbf{1} + (\mathsf{Tree}(N) \times N) \times (\mathsf{Forest}(N) \times N) \to N\end{array}}{\vdash \mathsf{Rec}_F(f_T, f_F)(forest(f)) = \begin{array}{c} f_F(\mathbf{case}(f, \mathsf{Inl}(x).\mathsf{In}(\star), \mathsf{Inr}(x).\langle\langle \pi_1 x, \mathsf{Rec}_T(f_T, f_F)(\pi_1 x)\rangle, \\ \langle \pi_2 x, \mathsf{Rec}_F(f_T, f_F)(\pi_2 x)\rangle\rangle)) : N\end{array}}$$

$$\frac{\begin{array}{l}\vdash t : N \times \mathsf{Forest}(N)\\ \vdash f_T \stackrel{\text{def}}{=} \lambda x : N \times (\mathsf{Forest}(N) \times N).\pi_1 x + \pi_2\pi_2 x : N \times (\mathsf{Forest}(N) \times N) \to N\\ \vdash f_F \stackrel{\text{def}}{=} \lambda x : \mathbf{1} + (\mathsf{Tree}(N) \times N) \times (\mathsf{Forest}(N) \times N).\\ \quad \mathbf{case}(x, \mathsf{Inl}(z).0, \mathsf{Inr}(z).\pi_2\pi_1 z + \pi_2\pi_2 z) : \mathbf{1} + (\mathsf{Tree}(N) \times N) \times (\mathsf{Forest}(N) \times N) \to N\end{array}}{\vdash \mathsf{Rec}_T(f_T, f_F)(tree(t)) = f_T(\langle \pi_1 t, \langle \pi_2 t, \mathsf{Rec}_F(f_T, f_F)(\pi_2 t)\rangle\rangle) : N}.$$

The conclusion is reducible to $\mathsf{Rec}_F(forest(f)) = \mathbf{case}(f, \mathsf{Inl}(x).0, \mathsf{Inr}(x).\mathsf{Rec}_T(\pi_2\pi_1 x) + \mathsf{Rec}_F(\pi_2\pi_2 x))$ *in the first case and* $\mathsf{Rec}_T(tree(t)) = \pi_1 t + \mathsf{Rec}_F(\pi_2 t)$ *in the second.*

3.2 Constructing Initial T-algebras

The inductive types as given in previous section are of course modeled by appropriate initial T-algebras. So the question is: given a concrete model of the *Calculus of Constructions* (or **ECC** or Martin-Löf's set theory) and an endofunctor F on the category within which the model is taken, how do we construct the initial F-algebra? The answer is given by the following theorem 3.5. This theorem generalizes a result in [AK79]. The proof of theorem 3.5 also generalizes that in [AK79].

Suppose T is an endofunctor on $\mathbf{C}$. A *T-algebra* is a pair $(A, TA \xrightarrow{f} A)$ where A is an object in $\mathbf{C}$ and f a morphism in $\mathbf{C}$. A *homomorphism* from $(A, TA \xrightarrow{f} A)$ to $(B, TB \xrightarrow{g} B)$ is a morphism $h : A \longrightarrow B$ such that $f; h = T(h); g$. An *initial T-algebra* is the initial object in the category of T-algebras. Dually, we can define (*terminal*) *T-coalgebras* (notation $(C \xrightarrow{h} TC, C)$). A *twisted homomorphism* from a T-coalgebra (h, C) to a T-algebra (A, f) is a morphism $l : C \longrightarrow A$ that satisfies the equation $l = h; T(l); f$, while a *cotwisted homomorphism* from (A, f) to (h, C) is a morphism $n : A \longrightarrow C$ such that $Tn = f; n; h$.

From now on suppose that $\mathbf{C}$ has filtered colimits (we assume it is equipped with specific colimiting cocones). We are ready to describe two important constructions.

Construction 1. The *chain induced by* a T-coalgebra $(a_{0,1}, A_0)$ is constructed as follows:

- The head of the chain is just $a_{0,1} : A_0 \longrightarrow TA_0 = A_1$.
- For a successor ordinal i, A_{i+1} is defined as TA_i and $a_{i,i+1}$ as $Ta_{i-1,i}$.

- For a limit ordinal γ, define A_γ to be the limit of the chain constructed so far. For $i \in \gamma$, $a_{i,\gamma} : A_i \longrightarrow A_\gamma$ is obtained from the cocone.

- For the same limit ordinal γ, $a_{\gamma,\gamma+1}$ is the unique mediating morphism as shown in the diagram:

$$\begin{tikzcd}
A_0 \arrow[r, "a_{0,1}"] \arrow[d, "a_{0,\gamma}"'] \arrow[dr, dashed] & A_1 \arrow[r, "a_{1,2}"] \arrow[dl, "a_{1,\gamma}"] \arrow[d, dashed, "a_{1,\gamma+1}"] & A_2 \arrow[r, "a_{2,3}"] \arrow[dl, dashed, "a_{2,\gamma+1}"] & A_3 \arrow[r, "a_{3,4}"] & \cdots \\
A_\gamma \arrow[r, "a_{\gamma,\gamma+1}"'] & A_{\gamma+1}. & & &
\end{tikzcd}$$

  The dotted diagram is obtained by applying the functor T to the cocone diagram constructed in the previous step. For limit ordinal $\alpha \in \gamma$, we define $a_{\alpha,\gamma+1}$ to be $a_{\alpha,\alpha+1}; Ta_{\alpha,\gamma}$. It is easily seen that we get a new cocone over the same diagram.

We say the chain *closes at* α if $a_{\alpha,\alpha+1}$ is an isomorphism and for no $\beta \in \alpha$ is $a_{\beta,\beta+1}$ an isomorphism. α is called the *closure ordinal.*

Lemma 3.4 *If the chain closes at α, then for any $\beta \geq \alpha$, $a_{\beta,\beta+1}$ is an isomorphism and for any two ordinals $\alpha \leq \beta < \gamma$, where γ is a limit ordinal, $a_{\beta,\gamma}$ is an isomorphism.*

Construction 2. Suppose (X, f) is a T-algebra and r is a twisted homomorphism from the T-coalgebra $(a_{0,1}, A_0)$ to (X, f). The *r-induced cocone* over the $(a_{0,1}, A_0)$-induced chain is constructed as follows:

- r_0 is the morphism r. Let $r_1 \stackrel{\text{def}}{=} T(r_0); f$. One verifies that $a_{0,1}; r_1 = a_{0,1}; T(r_0); f = r_0$.

- For a successor ordinal i, let $r_{i+1} \stackrel{\text{def}}{=} Tr_i; f$. We obtain

$$\begin{array}{rcl}
a_{i,i+1}; r_{i+1} & = & T(a_{i-1,i}); T(r_i); f \\
& = & T(a_{i-1,i}; r_i); f \\
& = & T(r_{i-1}); f \\
& = & r_i.
\end{array}$$

- Let γ be a limit ordinal. Because what we have constructed so far forms a cocone, we can define r_γ to be the unique mediating morphism.

- For the same limit ordinal γ, let $r_{\gamma+1} \stackrel{\text{def}}{=} T(r_\gamma); f$.

$$
\begin{tikzcd}
A_0 \arrow[r, "a_{0,1}"] \arrow[dr, "r_0"'] & TA_0 \arrow[r, "Ta_{0,1}"] \arrow[d, "r_1"'] \arrow[dr, "Tr_0"] & TA_1 \arrow[r, "Ta_{1,2}"] \arrow[d, "Tr_1"'] & TA_2 \arrow[r, "Ta_{2,3}"] \arrow[dl, "Tr_2"'] \arrow[d, "Ta_{2,\gamma}"'] \arrow[dr, "a_{3,\gamma}"] & \cdots \\
& X & TX \arrow[l, "f"] & TA_\gamma \arrow[l, "Tr_\gamma"] & A_\gamma \arrow[l, "a_{\gamma,\gamma+1}"]
\end{tikzcd}
$$

Applying T to the cocone diagram so far constructed, we have for each $i \in \gamma$

$$
\begin{aligned}
a_{i,\gamma}; a_{\gamma,\gamma+1}; r_{\gamma+1} &= a_{i,i+1}; a_{i+1,\gamma+1}; T(r_\gamma); f \\
&= a_{i,i+1}; T(a_{i,\gamma}); T(r_\gamma); f \\
&= a_{i,i+1}; T(a_{i,\gamma}; r_\gamma); f \\
&= a_{i,i+1}; T(r_i); f \\
&= a_{i,i+1}; r_{i+1} \\
&= a_{i,\gamma}; r_\gamma.
\end{aligned}
$$

By the uniqueness of the mediating morphism, we conclude that $a_{\gamma,\gamma+1}; r_{\gamma+1} = r_\gamma$.

Theorem 3.5 *Suppose $(a_{0,1}, A_0)$ and (X, f) are the same as in the above construction. There is a bijective correspondence between the class of twisted homomorphisms from $(a_{0,1}, A_0)$ to (X, f) and the class of twisted homomorphisms from $(a_{\alpha,\alpha+1}, A_\alpha)$ to (X, f) for every ordinal α.*

Proof: The construction *2* prescribes a function:

$$G_\alpha : \mathcal{H}[(a_{0,1}, A_0), (X, f)] \longrightarrow \mathcal{H}[(a_{\alpha,\alpha+1}, A_\alpha), (X, f)].$$

1. G_α is surjective. Suppose $s_\alpha : (a_{\alpha,\alpha+1}, A_\alpha) \longrightarrow (X, f)$ is a twisted homomorphism. Let $s_0 \stackrel{\text{def}}{=} a_{0,\alpha}; s_\alpha$. It is readily verified that $s_0 \in \mathcal{M}[(a_{0,1}, A_0), (X, f)]$. We only have to show that the last leg of the s_0-induced cocone is s_α. We will define a twisted homomorphism $s_i : (a_{i,i+1}, A_i) \longrightarrow (X, f)$ for each $i \in \alpha$ and prove by transfinite induction that (i) $s_i = a_{i,i+1}; s_{i+1}$ for successor ordinals and $s_i = a_{i,\gamma}; s_\gamma$ for limit ordinals and (ii) $a_{i,\alpha}; s_\alpha = s_i$.

- Define $s_1 \stackrel{\text{def}}{=} T(s_0); f$.

$$
\begin{aligned}
s_1 &= T(s_0); f \\
&= T(a_{0,\alpha}; s_\alpha); f \\
&= a_{1,\alpha+1}; Ts_\alpha; f \\
&= a_{1,\alpha}; s_\alpha, \\
a_{0,1}; s_1 &= a_{0,1}; T(a_{0,\alpha}; s_\alpha); f \\
&= a_{0,\alpha}; a_{\alpha,\alpha+1}; T(s_\alpha); f \\
&= a_{0,\alpha}; s_\alpha \\
&= s_0.
\end{aligned}
$$

- For i a successor ordinal, define $s_{i+1} \stackrel{\text{def}}{=} T(s_i); f$.

$$
\begin{aligned}
a_{i,i+1}; s_{i+1} &= a_{i,i+1}; T(s_i); f \\
&= a_{i,i+1}; T(a_{i,\alpha}; s_\alpha); f \\
&= a_{i,\alpha}; a_{\alpha,\alpha+1}; T(s_\alpha); f \\
&= a_{i,\alpha}; s_\alpha \\
&= s_i, \\
s_{i+1} &= T(s_i); f \\
&= T(a_{i,\alpha}; s_\alpha); f \\
&= a_{i+1,\alpha}; a_{\alpha,\alpha+1}; T(s_\alpha); f \\
&= a_{i+1,\alpha}; s_\alpha.
\end{aligned}
$$

- $\gamma \in \alpha$ is a limit ordinal. s_γ is defined as the mediating morphism. As $s_i = a_{i,\gamma}; s_\gamma$, we get $a_{i,\gamma}; a_{\gamma,\alpha}; s_\alpha = a_{i,\alpha}; s_\alpha = s_i = a_{i,\gamma}; s_\gamma$. It follows that $a_{\gamma,\alpha}; s_\alpha = s_\gamma$.

- For the limit ordinal γ, $s_{\gamma+1} \stackrel{\text{def}}{=} T(s_\gamma); f$. That $a_{\gamma,\gamma+1}; s_{\gamma+1} = s_\gamma$ is established in the same way as in the construction *2*. The proof of $s_{\gamma+1} = a_{\gamma+1}; s_\gamma$ is the same as in the second case.

If α is a successor ordinal, $T(s_{\alpha-1}); f = T(a_{\alpha-1,\alpha}; s_\alpha); f = a_{\alpha,\alpha+1}; T(s_\alpha); f = s_\alpha$. Otherwise what we have shown is that s_α is the mediating morphism. In both cases s_α is just what we would have defined.

2. G_α is injective. Let r_0 and t_0 be two different twisted homomorphisms from $(a_{0,1}, A_0)$ to (X, f). It is clear that $r_1 \neq t_1$ for otherwise $r_0 = a_{0,1}; r_1 = a_{0,1}; t_1 = t_0$. In general $r_i \neq t_i$ for successor ordinals. But then $r_\gamma \neq t_\gamma$ for limit ordinals because they are mediating morphisms induced by two different cocones. Hence $r_\alpha \neq t_\alpha$. ■

If **C** is also cofiltered complete, we can construct dually from a twisted homomorphism $r : (a_{0,1}, A_0) \longrightarrow (B_0, b_{0,1})$ the *r-induced cone* over the $(B_0, b_{1,0})$-induced cochain. It is easy to see that we get a commutative diagram:

$$
\begin{tikzcd}
A_0 \arrow[r, "a_{0,1}"] \arrow[d, "r"] \arrow[dr] & A_1 \arrow[r, "a_{1,2}"] \arrow[d, "Tr"] \arrow[dl] \arrow[dr] & A_2 \arrow[r, dashed] \arrow[d, "T^2 r"] \arrow[dl] & A_\alpha \arrow[r, "a_{\alpha,\alpha+1}"] \arrow[d] \arrow[dr] & A_{\alpha+1} \cdots \arrow[d] \arrow[dl] \\
B_0 & B_1 \arrow[l, "b_{1,0}"] & B_2 \arrow[l, "b_{2,1}"] & B_\alpha \arrow[l, dashed] & B_{\alpha+1} \cdots . \arrow[l, "b_{\alpha+1,\alpha}"]
\end{tikzcd}
$$

It is a simple consequence of the theorem that

$$\mathcal{H}[(a_{0,1}, A_0), (B_i, b_{i+1,i})] \simeq \mathcal{H}[(a_{j,j+1}, A_j), (B_0, b_{1,0})]$$

holds for all ordinals i, j. As we mentioned, theorem 3.5 generalizes nicely a result in [AK79] which is

Corollary 3.6 *Suppose* $\mathbf{C}$ *is filtered cocomplete with the initial object* $\varnothing$ *and* T *an endofunctor on* $\mathbf{C}$. *If the* $(\varnothing \xrightarrow{!} T\varnothing, \varnothing)$*-induced chain closes at* α*, then* $(T^{\alpha}\varnothing, (T^{\alpha}(!))^{-1})$ *is the initial* T*-algebra.*

3.3 Monotonic Functors on ω-Set

Let F :Set$\longrightarrow$Set be a monotonic functor; that is $F(A \hookrightarrow B)$ is an inclusion whenever $A \hookrightarrow B$ is. Then we can construct a sequence as shown in the diagram:

$$
\begin{tikzcd}
\varnothing \arrow[r, "!"] \arrow[d, hook] \arrow[dr] & F(\varnothing) \arrow[r, "F(!)"] \arrow[dl] \arrow[d] & F^2(\varnothing) \arrow[r, "F^2(!)"] \arrow[dl] & \cdots \\
F^{\beta}(\varnothing) \stackrel{\text{def}}{=} \cup_{\theta<\beta} F^{\theta}(\varnothing) \arrow[r, dashed, "f"'] & A. & &
\end{tikzcd}
$$

For a limit ordinal β, the colimit is defined by taking the union of the sets previously constructed. Suppose A is the vertex of a cocone. Then the mediating morphism f can be obviously defined.

If $M[X]$ is a type strictly positive with respect to X, then it can be interpreted as a monotonic functor on Set. This functor is bounded upwards for simple cardinality reason. So we can interpret $\mu X.M[X]$ as the initial T-algebra. This is the model given in [CPM90].

The situation in ω-Set is different. The problem is that $\cup_{\theta<\beta}F^{\theta}(\varnothing)$, when β is a limit ordinal, is not necessarily a colimit (ω-Set does not have all colimits). We still have the underlying function f as shown in the above diagram, but in general it is not guaranteed that there is a recursive function that tracks it.

As in the previous case, a strictly positive type constructor gives rise to a monotonic functor on ω-Set. What is unsatisfactory is that corollary 3.6 is no longer applicable. So the construction of an initial T-algebra from a monotonic functor on Set does not generalize to the case of ω-Set. In [Ore92], the author assumes that every bounded monotonic functor on ω-Set generates an initial T-algebra, but does not explain why.

Some bounded monotonic functors on ω-Set however do have initial T-algebras.

Definition 3.7 *We say an* ω*-set* $A = (A, \Vdash_A)$ *is* decidable *if the identity endomorphism* Id_A *is tracked by some recursive function defined* only *on the set* $\mathcal{R}(A) \stackrel{\text{def}}{=} \{n | n \Vdash_A a$ for some $a \in A\}$ *of its realizers (or equivalently,* Id_A *is tracked by the identity recursive function defined* only *on* $\mathcal{R}(A)$*).*

Definition 3.8 *We say a monotonic functor on ω-*Set *is* decidable *if it maps decidable ω-sets to decidable ω-sets. It is* continuous *if it is a bounded decidable monotonic functor whose closure ordinal is countable.*

Proposition 3.9 *Suppose $F : \omega$-*Set$\longrightarrow \omega$-Set *is a continuous functor. Then it has an initial T-algebra.*

Proof: The initial ω-set is decidable. By assumption, $F^i(\varnothing)$ is decidable. We need to show that $\cup_{i\in\alpha}F^i(\varnothing)$ is decidable for some limit ordinal $\alpha < \aleph_1$. As $F^i(\varnothing)$ is decidable, $Id_{F^i(\varnothing)}$ is tracked by some d_i that is defined only on $\mathcal{R}(F^i(\varnothing))$ for each $i \in \alpha$. $Id_{\cup_{i\in\alpha}F^i(\varnothing)}$ is then tracked by the following Turing machine:

- For any given $x \in \omega$, it imitates the first step of d_0 on x at the first stage.
- At the second stage, it imitates the first step of d_1 on x and the second step of d_0 on x.
- At the third stage, it imitates the first step of d_2, the second step of d_1 and the third step of d_0.
- And so on and so forth. As α is countable, this diagonal method gives rise to a Turing machine.

Clearly this machine stops only on inputs that are in $\mathcal{R}(\cup_{i\in\alpha}F^i(\varnothing))$. So $F^\alpha(\varnothing)$ is decidable. Next we must show that $F^\alpha(\varnothing)$ is the colimit. Suppose $\{f_i\}_{i\in\alpha}$ is a cocone and for each $i \in \alpha$, f_i is tracked by c_i. Then by the above analysis, for each $i \in \alpha$, f_i is also tracked by some e_i that is defined only on $\mathcal{R}(F^i(\varnothing))$. In view of what we have mentioned, we only have to find a recursive function that tracks the mediating function. This can be done by applying the above diagonal argument to $\{f_i\}_{i\in\alpha}$. ■

The continuous functors can be seen as the ω-Set counterparts of κ-continuous functors. Notice that if the ambient category is Per, the countability condition can be dropped.

Definition 3.10 *Suppose $X : Type_i \vdash \phi(X) : Type_i$. We say $\phi(X)$ is* rigidly positive *with respect to X if (i) $\phi = C$ such that the standard interpretation of C in ω-*Set *is a decidable ω-set or (ii) $\phi = X$ or (iii) $\phi = \phi_1 \otimes \phi_2$ such that both ϕ_1 and ϕ_2 are* rigidly positive *with respect to X where $\otimes$ is either $\times$ or $+$.*

Proposition 3.11 *Suppose $F(X)$ is a type and X occurs in it rigidly positively. Then the denotation of F in ω-*Set *(*Per*) is a continuous functor.*

It should be clear that, when restricted to the rigidly positive type constructors, the language defined in section 3.1 does have an interpretation in ω-Set.

4 Local Inductive Types

In some typed calculi or programming languages, say object oriented languages, there is a natural notion of subtyping relation $\preceq$. In such a language, we can define a type, inductively, that contains another type. We assume that coercions are implicit; that is $a : B$ whenever $A \preceq B$ and $a : A$. The following rules describe how the *local inductive types* are formed and manipulated.

Formation

$$\frac{\Gamma \vdash B : \mathit{Type} \quad \Gamma, X : \mathit{Type} \vdash M : \mathit{Type} \quad \Gamma \vdash B \preceq M[B/X] : \mathit{Type}}{\Gamma \vdash B \preceq \mu X.[B \uparrow M] : \mathit{Type}}$$

where X appears strictly positively in M. The same restriction applies to other rules.

Introduction

$$\frac{\Gamma \vdash N : M[\mu X.[B \uparrow M]/X]}{\Gamma \vdash \mathsf{intro}_\mu(N) : \mu X.[B \uparrow M]}$$

Elimination

$$\frac{\begin{array}{l} \Gamma \vdash B : \mathit{Type} \quad \Gamma, X : \mathit{Type} \vdash M : \mathit{Type} \quad \Gamma \vdash B \preceq M[B/X] : \mathit{Type} \\ \Gamma \vdash h : \Pi x : B.C \quad \Gamma \vdash f : \Pi z : M[\Sigma x : \mu.C/X].C[\mathsf{intro}_\mu(M^\star(\pi_1)z)] \\ \Gamma \vdash \lambda x : B.f(M^\star(\langle \lambda x : B.x, h\rangle)x) = h : \Pi x : B.C \end{array}}{\Gamma \vdash \mathsf{Rec}_\mu(f \uparrow h) : \Pi x : \mu X.[B \uparrow M].C}$$

Computation

$$\frac{\begin{array}{l} \Gamma \vdash B : \mathit{Type} \quad \Gamma, X : \mathit{Type} \vdash M : \mathit{Type} \quad \Gamma \vdash B \preceq M[B/X] : \mathit{Type} \\ \Gamma \vdash h : \Pi x : B.C \quad \Gamma \vdash f : \Pi z : M[\Sigma x : \mu.C/X].C[\mathsf{intro}_\mu(M^\star(\pi_1)z)] \\ \Gamma \vdash \lambda x : B.f(M^\star(\langle \lambda x : B.x, h\rangle)x) = h : \Pi x : B.C \\ \Gamma \vdash a : B \end{array}}{\Gamma \vdash \mathsf{Rec}_\mu(f \uparrow h)(a) = h(a) : C[a]}$$

$$\frac{\begin{array}{l} \Gamma \vdash B : \mathit{Type} \quad \Gamma, X : \mathit{Type} \vdash M : \mathit{Type} \quad \Gamma \vdash B \preceq M[B/X] : \mathit{Type} \\ \Gamma \vdash h : \Pi x : B.C \quad \Gamma \vdash f : \Pi z : M[\Sigma x : \mu.C/X].C[\mathsf{intro}_\mu(M^\star(\pi_1)z)] \\ \Gamma \vdash \lambda x : B.f(M^\star(\langle \lambda x : B.x, h\rangle)x) = h : \Pi x : B.C \\ \Gamma \vdash a : M[\mu X.[B \uparrow M]] \end{array}}{\Gamma \vdash \mathsf{Rec}_\mu(f \uparrow h)(\mathsf{intro}_\mu(a)) = f(M^\star(\mathsf{Rec}_\mu(f \uparrow h))(a)) : C[\mathsf{intro}_\mu(M^\star(\pi_1)a)]}$$

The formulation of the above rules is motivated by theorem 3.5. The following diagram helps to understand the above rules:

$$\begin{tikzcd}
B \arrow[r, "\lambda x : B.x"] \arrow[d, "\langle \lambda x : B.x, h\rangle"'] & M[B] \arrow[d, "M^\star(\langle \lambda x : B.x, h\rangle)"] \\
\Sigma x : \mu.C & M[\Sigma x : \mu.C]. \arrow[l, "\langle M^\star(\pi_1); \mathsf{intro}_\mu, f\rangle"]
\end{tikzcd}$$

The local inductive types can also be formulated in the traditional style. Without losing any generality, we explain only the rules for local W-types.

Formation

$$\frac{\Gamma, x : A \vdash B : Type \quad \Gamma \vdash D : Type}{\Gamma \vdash D \preceq W_D x : A.B : Type}$$

Introduction

$$\frac{\Gamma \vdash a : A \quad \Gamma \vdash b : B[a] \to W_D x : A.B}{\Gamma \vdash \mathbf{sup}(a, b) : W_D x : A.B}$$

Elimination

$$\frac{\begin{array}{l}\Gamma, z : W_D x : A.B \vdash C : Type \\ \Gamma \vdash f_0 : \Pi z : D.C \\ \Gamma \vdash f_1 : \Pi x : A.\Pi y : (B \to W_D x : A.B).\Pi h : (\Pi b : B.C[y(b)]).C[\mathbf{sup}(x, y)]\end{array}}{\Gamma \vdash \mathsf{rec}_{W_D x A.B}(f_0, f_1) : \Pi z : (W_D x : A.B).C}$$

Computation

$$\frac{\begin{array}{l}\Gamma, z : W_D x : A.B \vdash C : Type \\ \Gamma \vdash f_0 : \Pi z : D.C \\ \Gamma \vdash f_1 : \Pi x : A.\Pi y : (B \to W_D x : A.B).\Pi h : (\Pi b : B.C[y(b)]).C[\mathbf{sup}(x, y)] \\ \Gamma \vdash d : D\end{array}}{\Gamma \vdash \mathsf{rec}_{W_D x A.B}(f_0, f_1)(d) = f_0(d) : C[d]}$$

$$\frac{\begin{array}{l}\Gamma, z : W_D x : A.B \vdash C : Type \\ \Gamma \vdash f_0 : \Pi z : D.C \\ \Gamma \vdash f_1 : \Pi x : A.\Pi y : (B \to W_D x : A.B).\Pi h : (\Pi b : B.C[y(b)]).C[\mathbf{sup}(x, y)] \\ \Gamma \vdash a : A \quad \Gamma \vdash b : B[a] \to W_D x : A.B\end{array}}{\Gamma \vdash \mathsf{rec}_{W_D x A.B}(f_0, f_1)(\mathbf{sup}(a, b)) = f_1(a, b, \lambda x : B[a].\mathsf{rec}_{W_D x A.B}(f_0, f_1)(b(x))) : C[\mathbf{sup}(a, b)]}$$

The ω-SET model as described in § 2.3 can be extended to interpret the local W-types. For $\alpha \in |\Gamma|$, the rule set R_α is defined as the following set:

$$\left\{ \frac{}{d} \;\middle|\; d \in |D| \right\} \cup \left\{ \frac{range(b)}{\langle n_{sup}, a, b \rangle} \;\middle|\; \begin{array}{l} a \in |A(\alpha)| \wedge \\ b \in |B(\alpha, a)| \to V_\kappa \end{array} \right\}.$$

The effective procedure for getting $\omega(\mathcal{I}(R_\alpha))$ remains almost unchanged except that this time we take ω^0 to be D. The graph F_α is

$$Graph(f_0(\alpha)) \cup \left\{ \frac{\{\langle b(x), h(x) \rangle \mid x \in |B(\alpha, a)|\}}{\langle \langle n_{sup}, a, b \rangle, f_1(\alpha)(a, b, h) \rangle} \;\middle|\; \begin{array}{l} a \in |A(\alpha)| \wedge \\ b \in |B(\alpha, a) \to \omega(\mathcal{I}(R_\alpha))| \wedge \\ h \in |\pi(B(\alpha, a), C(\alpha, b(\_)))| \end{array} \right\}.$$

Assuming the elements are distinguished by a code. A recursive function that tracks $\mathcal{I}(F_\alpha)$ can be found similarly. The only difference is that there should be a branching programme that will perform different actions depending on whether the input datum is in D or not.

5 Conclusion and Further Work

There are two ways to define an inductive set: one is to take the smallest closure of a rule set and the other is to take the least fixed point of a bounded monotonic functor. Although equivalent in classical set theory, the former is weaker in a constructive setting. In this paper, we have explained how to interpret the generalized inductive types in ω-SET by effectivizing the notion of rule sets. We have also noticed that the notion of κ-continuous functors on SET can not be transplanted to ω-SET without considerable restriction. At the moment, we do not know if the continuous functors are a general enough definition. The interpretation given in section 2.3 involves a huge amount of encoding. Is there a civilized or categorical description? Is it possible to give an internal account of what is done in section 2.3? These questions are closely related to a much harder one: what are the endofunctors on ω-SET that have initial T-algebras?

6 Acknowledgement

The material in this paper is taken from the author's Ph.D thesis ([Fu92d]). 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 support from the British Council, the State Education Commission of China and the late Yu-Gang Bau and the CLICS-II project is gratefully acknowledged.

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.

[AK79] J. Adámek and V. Koubek. Least fixed point of a functor. *Journal of Computer Systems*, 19:163–178, 1979.

[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.

[Ber89] S. Berardi. *Non-conservativity of Coquand's Calculus With Respect To Higher-order Intuitionistic Logics*, July 1989.

[CH88] T. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988.

[Chu40] A. Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

[Coq86] T. Coquand. An analysis of Girard's paradox. In *Proceedings of the First Symposium on Logic in Computer Science*, pages 227–236. IEEE Computer Science Press, June 1986.

[Coq90] T. Coquand. Mathematical investigations of a calculus of constructions. In P. Odifreddi, editor, *Logic and Computer Science*, pages 91–122. Academic Press, 1990.

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

[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.

[Fu92a] Yuxi Fu. *A Note on Topos Models.* Department of Computer Science, University of Manchester, July 1992.

[Fu92b] Yuxi Fu. *Recursion and Universes in* ω-SET. Department of Computer Science, University of Manchester, July 1992.

[Fu92c] Yuxi Fu. *Recursive Models of General Inductive Types.* Department of Computer Science, University of Manchester, July 1992.

[Fu92d] Yuxi Fu. *Some Semantic Issues In Type Theory.* PhD thesis, Department of Computer Science, University of Manchester, May 1992.

[Fu93a] Yuxi Fu. *Categorical Properties of Logical Frameworks.* Department of Computer Science, University of Manchester, February 1993.

[Fu93b] Yuxi Fu. *Encodings in Polymorphism, Revisited.* Department of Computer Science, University of Manchester, January 1993.

[Fu93c] Yuxi Fu. *Logical Type Theory.* Department of Computer Science, University of Manchester, May 1993.

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

[How80] W. Howard. The formulae-as-type notion of construction. In J. Seldin and J. Hindley, editors, *To H. B. Curry: Essays in Combinatory Logic, Lambda Calculus and Formalism.* Academic Press, 1980.

[Lev79] A. Levy. *Basic Set Theory.* Springer-Verlag, 1979.

[Luo90a] Z. Luo. *An Extended Calculus of Constructions.* PhD thesis, LFCS, University of Edinburgh, Edinburgh, July 1990.

[Luo90b] Z. Luo. *A Problem of Adequacy: conservativity of calculus of constructions over higher-order logic.* LFCS, University of Edinburgh, August 1990.

[Luo91] Z. Luo. A higher-order calculus and theory abstraction. *Information and Computation*, 90(1):107–137, January 1991.

[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.

[Men91] 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.

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

[Pit87] A. Pitts. Polymorphism is set-theoretic, constructively. In D. Pitt, A.Poigné, and D. Rydeheard, editors, *Category Theory and Computer Science, LNCS 283*, pages 12–39, Edinburgh, U.K., September 1987. Springer Verlag.

[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.

[PS89] K. Petersson and D. Synek. A set constructor for inductive sets in martin-löf's type theory. In D. Pitt, D. Rydeheard, P. Dybier, A. Pitts, and A. Poigné, editors, *Category Theory and Computer Science, LNCS 389*, pages 128–140, Manchester, September 1989. Springer Verlag.

[RW12] B. Russell and A. Whitehead. *The Principia Mathematica*, volume 1. Cambridge University Press, 1912.

[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.