Vol.16 No.2 J. Comput. Sci. & Technol. Mar. 2001

Semantics of Constructions (Ⅱ) — The Initial Algebraic Approach

FU Yuxi (傅育熙)

Department of Computer Science, Shanghai Jiao Tong University, Shanghai 200030, P.R. China

E-mail: fu-yx@cs.sjtu.edu.cn

Received January 11, 1999; revised April 14, 2000.

Abstract Inductive types can be formulated by incorporating the idea of initial T-algebra. The interpretation of an inductive type of this kind boils down to finding out the initial T-algebra defined by the inductive type. In this paper the issue in the semantic domain of omega sets is examined. Based on the semantic results, a new class of inductive types, that of local inductive types, is proposed.

Keywords type theory, inductive type, ω-set, T-algebra

1 Introduction

Inductive types play an important role in type theory. In Martin-Löf type theory[1], all types are inductive types. The idea of inductive types is very much set theoretical. Recursion rules for inductive types, for example, are type theoretical version of set recursion in classical set theory. As a matter of fact, inductive types in Martin-Löf type theory have set theoretical semantics in classical set theory. The situation is different in the calculus of constructions[2]. As the language is impredicative, a set theoretical model is impossible. But impredicativity does not rule out a constructive set theoretical semantics. In [3] an ω-set semantics is given for generalized inductive types in the calculus of constructions. The model is obtained by effectivizing, so to speak, the classical models of inductive types of Martin-Löf type theory.

There is another way of formalizing the ideas embodied in the generalized inductive types by using initial T-algebras. 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. The initial algebraic approach is studied in [4]. Unfortunately the semantics defined in [4] is not sound. The mistake made in [4] is that the author assumed that a continuous endofunctor on the category of ω-sets always has an initial T-algebra. It is true that a continuous endofunctor on the category of sets always has an initial T-algebra because there is a unique mediating functor from a T-algebra to the initial T-algebra. But this functor is not always effective; there is not necessarily a computable function that tracks the mediating functor. This says that the mediating functor in case of ω-sets does not necessarily exist. The purpose of this paper is to give a more formal treatment of the initial algebraic semantics in the category of ω-sets.

In this paper the rules for mutually dependent inductive types are first spelled out. Two examples are given to illustrate the usage of these rules. Then a general method to construct interpretations of these types is illustrated, based on a generalization of a well-known result. The construction then motivates the definition of a new class of inductive types to be described in the next section.

This work is funded by the National Natural Science Foundation of China (No.69973030). It is also supported by BASICS, Center of Basic Studies in Computing Science, sponsored by Shanghai Education Committee. BASICS is affiliated to the Department of Computer Science at Shanghai Jiao Tong University.

This paper is a sequel to [3]. It is not self-contained. The reader is referred to [3] for background information on the ω-set semantics of type theory.

2 The Language

Definition 2.1. *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 X does not occur in $\phi(X)$ or $\phi(X) = \Pi x_1 : K_1.\cdots\Pi x_m : K_m.X$ $(m \in \omega)$ such that X does not occur in any of $K_1, \ldots, K_m$.*

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

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

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

Recursion

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

where $\vec{\pi_1} \overset{\text{def}}{=} \lambda w_1 : (\Sigma x_1 : \mu^i_1.C_1).\pi_1 w_1, \ldots, \lambda w_n : (\Sigma x_n : \mu^i_n.C_n).\pi_1 w_n$. $M^\star_j$, $j \in [1, \ldots, n]$, 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

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

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

Extensionality

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

where $\langle \vec{id}, g\rangle \overset{\text{def}}{=} \lambda z : \mu_n^i.\langle z, g_n z\rangle, \ldots, \lambda z : \mu_n^i.\langle z, g_n z\rangle$.

Fig.1 helps to understand the rules concerning the inductive types.

$$\begin{tikzcd}
M_j[\mu_1, \ldots, \mu_n] \arrow[r, "M_j^\star(\langle id, rec_1^\mu(\vec{f})\rangle, \ldots, \langle id, rec_n^\mu(\vec{f})\rangle)"] \arrow[d, "intro_j^\mu"'] & M_j[\Sigma x_1 : \mu_1.C_1, \ldots, \Sigma x_n : \mu_n.C_n] \arrow[d, "\langle M_j^\star(\vec{\pi}_1); intro_j^\mu, f_j\rangle"] \\
\mu_j \arrow[r, "\langle id, rec_j^\mu(\vec{f})\rangle"'] & \Sigma x_j : \mu_j.C_j
\end{tikzcd}$$

Fig.1

Example 2.2. The best-known mutually dependent inductive type is the data type of trees and forests. Using Hagino's notation[5], 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):

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

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

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

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

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

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

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

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

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

Example 2.3. (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:

$$\frac{\begin{array}{l}\vdash t : N \times Forest(N)\\ \vdash f_T \overset{\text{def}}{=} \lambda x : N \times (Forest(N) \times N).\pi_1 x + \pi_2\pi_2 x : N \times (Forest(N) \times N) \to N\\ \vdash f_F \overset{\text{def}}{=} \lambda x : \mathbf{1} + (Tree(N) \times N) \times (Forest(N) \times N).\\ \quad \mathbf{case}(x, inl(z).0, inr(z).\pi_2\pi_1 z + \pi_2\pi_2 z) : \mathbf{1} + (Tree(N) \times N) \times (Forest(N) \times N) \to N\end{array}}{\vdash rec_F(f_T, f_F)(forest(f)) = \begin{array}{l} f_F(\mathbf{case}(f, inl(x).In(\star), inr(x).\langle\langle \pi_1 x, rec_T(f_T, f_F)(\pi_1 x)\rangle,\\ \langle \pi_2 x, rec_F(f_T, f_F)(\pi_2 x)\rangle\rangle)) : N\end{array}}$$

$$\frac{\begin{array}{l}\vdash t : N \times Forest(N)\\ \vdash f_T \overset{\text{def}}{=} \lambda x : N \times (Forest(N) \times N).\pi_1 x + \pi_2\pi_2 x : N \times (Forest(N) \times N) \to N\\ \vdash f_F \overset{\text{def}}{=} \lambda x : \mathbf{1} + (Tree(N) \times N) \times (Forest(N) \times N).\\ \quad \mathbf{case}(x, inl(z).0, inr(z).\pi_2\pi_1 z + \pi_2\pi_2 z) : \mathbf{1} + (Tree(N) \times N) \times (Forest(N) \times N) \to N\end{array}}{\vdash rec_T(f_T, f_F)(tree(t)) = f_T(\langle \pi_1 t, \langle \pi_2 t, rec_F(f_T, f_F)(\pi_2 t)\rangle\rangle) : N}$$

The conclusion is reducible to $rec_F(forest(f)) = \mathbf{case}(f, inl(x).0, inr(x).rec_T(\pi_2\pi_1 x) + rec_F(\pi_2\pi_2 x))$ in the first case and $rec_T(tree(t)) = \pi_1 t + rec_F(\pi_2 t)$ in the second.

3 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 3.2. This theorem generalizes a result in [6]. The proof of Theorem 3.2 also generalizes that in [6].

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. Duly, 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 Fig.2:

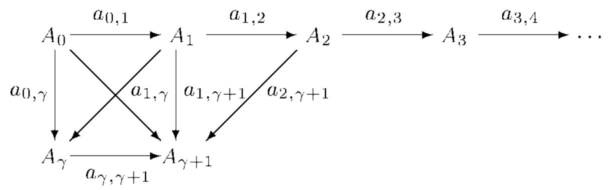
$$\begin{tikzcd}
A_0 \arrow[r, "a_{0,1}"] \arrow[d, "a_{0,\gamma}"'] \arrow[dr] & A_1 \arrow[r, "a_{1,2}"] \arrow[dl, "a_{1,\gamma}"] \arrow[d, "a_{1,\gamma+1}"] & A_2 \arrow[r, "a_{2,3}"] \arrow[dl, "a_{2,\gamma+1}"] & A_3 \arrow[r, "a_{3,4}"] & \cdots \\
A_\gamma \arrow[r, "a_{\gamma,\gamma+1}"'] &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

Fig.2

Here $a_{\alpha+1,\gamma+1}$ is defined as $Ta_{\alpha,\gamma}$. 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 $\beta \in \alpha$, $a_{\beta,\beta+1}$ is an isomorphism. α is called the closure ordinal.

Lemma 3.1. *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 $r_1 \overset{\text{def}}{=} T(r_0); f$. We can verify $a_{0,1}; r_1 = a_{0,1}; T(r_0); f = r_0$.

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

$$a_{i,i+1}; r_{i+1} = T(a_{i-1,i}); T(r_i); f = T(a_{i-1,i}; r_i); f = T(r_{i-1}); f = r_i$$

• 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 $r_{\gamma+1} \overset{\text{def}}{=} T(r_\gamma); f$ as shown in Fig.3.

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

Fig.3

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 3.2. *Suppose $(a_{0,1}, A_0)$ and (X, f) are the same as those 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 $s_0 \overset{\text{def}}{=} a_{0,\alpha}; s_\alpha$. 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 $s_1 \overset{\text{def}}{=} T(s_0); f$.

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

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

$$a_{i,i+1}; s_{i+1} = a_{i,i+1}; T(s_i); f = a_{i,i+1}; T(a_{i,\alpha}; s_\alpha); f = a_{i,\alpha}; a_{\alpha,\alpha+1}; T(s_\alpha); f = a_{i,\alpha}; s_\alpha = s_i$$
$$s_{i+1} = T(s_i); f = T(a_{i,\alpha}; s_\alpha); f = a_{i+1,\alpha}; a_{\alpha,\alpha+1}; T(s_\alpha); f = a_{i+1,\alpha}; s_\alpha$$

• $\gamma \in \alpha$ 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, $s_{\gamma+1} \overset{\text{def}}{=} T(s_\gamma); f$. That $a_{\gamma,\gamma+1}; s_{\gamma+1} = s_\gamma$ is established in the same way as in 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 $\mathbf{C}$ is also cofiltered complete, we can construct duly the *r-induced cone* over the $(B_0, b_{1,0})$-induced cochain from a twisted homomorphism $r : (a_{0,1}, A_0) \longrightarrow (B_0, b_{0,1})$. It is easy to see that we get a commutative diagram as shown in Fig.4:

$$\begin{tikzcd}
A_0 \arrow[r, "a_{0,1}"] \arrow[d, "r"] \arrow[dr] & A_1 \arrow[r, "a_{1,2}"] \arrow[d, "Tr"] \arrow[dl] \arrow[dr] & A_2 \arrow[r] \arrow[d, "T^2r"] \arrow[dl] & A_\alpha \arrow[r, "a_{\alpha,\alpha+1}"] \arrow[d, "T^\alpha r"] \arrow[dr] & A_{\alpha+1} \cdots \arrow[d, "T^{\alpha+1}r"] \arrow[dl] \\
B_0 \arrow[r, "b_{1,0}"'] & B_1 & B_2 \arrow[l, "b_{2,1}"] & B_\alpha \arrow[l] & B_{\alpha+1} \cdots \arrow[l, "b_{\alpha+1,\alpha}"]
\end{tikzcd}$$

Fig.4

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 3.2 generalizes nicely a result in [6] which is

Corollary 3.3. *Suppose* $\mathbf{C}$ *is filtered cocomplete with the initial object* $\emptyset$ *and* T *an endofunctor on* $\mathbf{C}$. *If the* $(\emptyset \xrightarrow{!} T\emptyset, \emptyset)$*-induced chain closes at* α, *then* $(T^\alpha\emptyset, (T^\alpha(!))^{-1})$ *is the initial* T*-algebra.*

4 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 Fig.5:

$$\begin{tikzcd}
\emptyset \arrow[r, "!"] \arrow[d] \arrow[dr] & F\emptyset \arrow[r, "F(!)"] \arrow[d] \arrow[dl] & F^2\emptyset \arrow[r, "F^2(!)"] \arrow[dl] & F^3\emptyset \arrow[r, "F^3(!)"] & \cdots \\
F^\beta(\emptyset) \overset{\text{def}}{=} \cup_{\theta<\beta} F^\theta(\emptyset) \arrow[r, "f"'] & A & & &
\end{tikzcd}$$

Fig.5

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 [7].

The situation in ω-Set is different. The problem is that $\cup_{\theta<\beta} F^\theta(\emptyset)$, 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 Fig.5, 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 3.3 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 [4], 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 4.1. *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 $\mathcal{R}(A) \overset{\text{def}}{=} \{n | n \vdash_A a$ 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 4.2. *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 4.3. *Suppose $F : \omega\text{-Set} \longrightarrow \omega\text{-Set}$ is a continuous functor. Then it has an initial T-algebra.*

Proof. The initial ω-set is decidable. By assumption, $F^i(\emptyset)$ is decidable. We need to show that $\cup_{i\in\alpha} F^i(\emptyset)$ is decidable for some limit ordinal $\alpha < \aleph_1$. As $F^i(\emptyset)$ is decidable, $Id_{F^i(\emptyset)}$ is tracked by some d_i that is defined only on $\mathcal{R}(F^i(\emptyset))$ for each $i \in \alpha$. $Id_{\cup_{i\in\alpha} F^i(\emptyset)}$ 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 $\mathcal{R}(\cup_{i\in\alpha} F^i(\emptyset))$. So $F^\alpha(\emptyset)$ is decidable. Next we must show that $F^\alpha(\emptyset)$ 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 $\mathcal{R}(F^i(\emptyset))$. 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 4.4. *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 4.5. *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 2 does have an interpretation in ω-Set.

5 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 local inductive types are formed and manipulated.

Formation

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

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

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

Elimination

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

Computation

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

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

The formulation of the above rules is motivated by Theorem 3.2. Fig.6 helps to understand the above rules:

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

Fig.6

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

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

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

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

Computation

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

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

The ω-Set model as described in [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

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

Assume 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.

6 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. 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 cannot 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 [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 [3]? These questions are closely related to a much harder one: what are the endofunctors on ω-Set that have initial T-algebras?

References

[1] Nordström B, Petersson K, Smith J. Programming in Martin-Löf's type theory — An introduction. In *International Series of Monographs on Computer Science 7,* Oxford University Press, 1990.

[2] Coquand T, Huet G. The calculus of constructions. *Information and Computation,* 1998, 76(1): 95–120.

[3] Fu Y. Semantics of constructions (I) — The traditional approach. *Journal of Computer Science and Technology*, 2001, 16(1): 13–24.

[4] Ore C. The extended calculus of constructions (ECC) with inductive types. *Information and Computation*, 1992, 99(1): 231–264.

[5] Hagino S. A typed lambda calculus with categorical type constructions. In *Category Theory in Computer Science,* Lecture Notes in Computer Science 283, Springer, 1987, pp.140–157.

[6] Adámek J, Koubek V. Least fixed point of a functor. *Journal of Computer Systems*, 1979, 19(1): 163–178.

[7] Coquand T, Paulin-Mohring C. Inductively Defined Types. In *COLOG'88, Lecture Notes in Computer Science 417,* Springer-Verlag, 1990, pp.50–66.

For the biography of **FU Yuxi**, please refer to p.24, No.1, Vol.16 of this journal.