Induction by Coinduction and Control Operators in Call-by-Name

Yoshihiko Kakutani
Department of Information Science, University of Tokyo, Japan
kakutani@is.s.u-tokyo.ac.jp

Daisuke Kimura
National Institute of Informatics, Japan
kmr@nii.ac.jp

This paper studies emulation of induction by coinduction in a call-by-name language with control operators. Since it is known that call-by-name programming languages with control operators cannot have general initial algebras, interaction of induction and control operators is often restricted to effect-free functions. We show that some class of such restricted inductive types can be derived from full coinductive types by the power of control operators. As a typical example of our results, the type of natural numbers is represented by the type of streams. The underlying idea is a counterpart of the fact that some coinductive types can be expressed by inductive types in call-by-name pure language without side-effects.

1 Introduction

In programming languages, natural numbers are one of the most important data types. There are many studies on natural numbers both in the theoretic field and in the practical field. An important feature of natural numbers is the induction principle. In a set-theoretic approach, the set of natural numbers $\mathbb{N}$ is defined inductively. It means that a function H from $\mathbb{N}$ to a set A is defined by two equations

$$H0 = M$$
$$H(n{+}1) = F(Hn)$$

where $M \in A$ and $F : A \to A$. A type of natural numbers can be introduced to the simply typed call-by-name λ-calculus, which is core of functional programming languages, in a similar way. Such a characterization can be regarded as an initial algebra. For example, [9] is a typical study of this approach.

In a cartesian closed category with coproducts, which is a model of the λ-calculus, it is known that a data type of infinite streams is the dual of the data type of natural numbers though streams are less useful than natural numbers. Intuitively, a stream is an infinite sequence indexed by natural numbers. This intuition is formalized as an isomorphism of the type of streams and the function type from natural numbers to some data type.

The type of natural numbers can be generalized to inductive types and the type of streams can be generalized to coinductive types. Semantically, an inductive type is characterized as an initial algebra. The universality of an initial algebra enables us to define a function on the data type inductively. Coinductive types corresponds to final coalgebras, the dual notion of initial algebras. In the call-by-name λ-calculus without side-effects, a coinductive type in some class can be expressed as a function type from an inductive type. The construction of streams from natural numbers is an example of this method.

The $\lambda\mu$-calculus, an extension of the λ-calculus with first-class continuations, was introduced by Parigot in [10]. Selinger has provided a categorical model of the call-by-name $\lambda\mu$-calculus in [12]. It has been shown in [13] that control operators cannot coexist with an initial object. Since an initial

U. de'Liguoro and A. Saurin (Eds.):
Control Operators and their Semantics 2013 (COS'13)
EPTCS 127, 2013, pp. 101–112, doi:10.4204/EPTCS.127.7

object is a special case of initial algebras, existence of general initial algebras is not allowed in the $\lambda\mu$-calculus. In such a situation, effect-free functions looks important. In the second-order $\lambda\mu$-calculus, the parametricity restricted to focal functions was proposed by [4]. Focal functions are a kind of effect-free functions and their detailed analyses are found in [12]. In this paper, we propose a reasoning about focal restriction of initial algebras.

Unlike the case of the pure λ-calculus, the $\lambda\mu$-calculus may not have general initial algebras. Hence, coinductive types are not expressed by inductive types. Instead, general coinductive types can exist in the $\lambda\mu$-calculus and focally-restricted inductive types can be expressed by function types from coinductive types. This is the main result of this work. Such contravariance between the $\lambda\mu$-calculus and the λ-calculus derived from the CPS semantics. Selinger's CPS transformation from the call-by-name $\lambda\mu$-calculus to the simply typed λ-calculus sends product types to coproduct types and premonoidal disjunction types to product types. This CPS transformation is based on the categorical semantics of the $\lambda\mu$-calculus and slightly different from Plotkin's original CPS transformation [11]. Barthe and Uustalu have studied a CPS transformation with inductive types and coinductive types in [2], but their transformation does not deal with the universality.

The construction of this paper is the following. In Section 2, we prepare the target calculus. We discuss natural numbers and streams in Section 3. In Section 4, results given in Section 3 are generalized. We show some instances of our results in Section 5.

2 CBN Calculus with First-Class Continuations

First, we introduce the base calculus, the call-by-name $\lambda\mu$-calculus. Our version of the $\lambda\mu$-calculus is based on Selinger's [12].

Definition 1. Types A, terms M of the call-by-name $\lambda\mu$-calculus are defined by

$$A ::= \tau \mid A \to A \mid \top \mid A \times A \mid \bot \mid A \vee A$$
$$M ::= c \mid x \mid \lambda x^A.M \mid MM \mid \langle\rangle \mid \langle M, M\rangle \mid \pi_1 M \mid \pi_2 M \mid \mu a^A.M \mid aM \mid \mu(a^A, a^A).M \mid [a,a]M$$

where τ, c, x, and a range over type constants, constants, variables and control variables, respectively. The constructors λ and μ binds variables in the usual way. Superscripted types may be omitted. A judgment has the form

$$x_1 : B_1, \ldots, x_n : B_n \vdash M : A \mid a_1 : A_1, \ldots, a_m : A_m$$

with two kinds of typing environments. Typing rules are given in Table 1, where the exchange rule on environments is implicitly assumed. The equality is the smallest congruence relation including axioms in Table 2, where an axiom $M = N$ means that $\Gamma \vdash M : A \mid \Delta$ and $\Gamma \vdash N : A \mid \Delta$ are equal when both judgments are deducible. A substitution $\{C/a-\}$ means a recursive replacement of aM by $C[M\{C/a-\}]$ and $[a_1, a_2]M$ by $C[\mu a.[a_1, a_2]M\{C/a-\}]$ when a is either a_1 or a_2.

Any well-typed term of the simply typed λ-calculus without disjunction types can be derived straightforwardly in the $\lambda\mu$-calculus if we forget the right-hand side typing environments. In fact, while the λ-calculus corresponds to the intuitionistic logic, $\lambda\mu$-calculus corresponds to the classical logic. In such Curry-Howard correspondence, types in a left-hand side environment combine conjunctively and types in a right-hand side environment combine disjunctively.

Table 1: Typing rules of $\lambda\mu$

$$\frac{}{\Gamma \vdash c^A : A \mid \Delta} \qquad \frac{}{\Gamma, x:A \vdash x : A \mid \Delta} \qquad \frac{}{\Gamma \vdash \langle\rangle : \top \mid \Delta}$$

$$\frac{\Gamma, x:B \vdash M : A \mid \Delta}{\Gamma \vdash \lambda x^B.M : B \to A \mid \Delta} \qquad \frac{\Gamma \vdash M : B \to A \mid \Delta \quad \Gamma \vdash N : B \mid \Delta}{\Gamma \vdash MN : A \mid \Delta}$$

$$\frac{\Gamma \vdash M_1 : A_1 \mid \Delta \quad \Gamma \vdash M_2 : A_2 \mid \Delta}{\Gamma \vdash \langle M_1, M_2\rangle : A_1 \times A_2 \mid \Delta} \qquad \frac{\Gamma \vdash M : A_1 \times A_2 \mid \Delta}{\Gamma \vdash \pi_j M : A_j \mid \Delta}$$

$$\frac{\Gamma \vdash M : \bot \mid a:A, \Delta}{\Gamma \vdash \mu a^A.M : A \mid \Delta} \qquad \frac{\Gamma \vdash M : A \mid a:A, \Delta}{\Gamma \vdash aM : \bot \mid a:A, \Delta}$$

$$\frac{\Gamma \vdash M : \bot \mid a_1:A_1, a_2:A_2, \Delta}{\Gamma \vdash \mu(a_1^{A_1}, a_2^{A_2}).M : A_1 \vee A_2 \mid \Delta} \qquad \frac{\Gamma \vdash M : A_1 \vee A_2 \mid a_1:A_1, a_2:A_2, \Delta}{\Gamma \vdash [a_1, a_2]M : \bot \mid a_1:A_1, a_2:A_2, \Delta}$$

An equational extension of the call-by-name $\lambda\mu$-calculus is called a $\lambda\mu$-theory. (We also call an extension of the simply typed λ-calculus a λ-theory.) A $\lambda\mu$-theory is needed for considering natural numbers or other structures, but we do not strictly distinguish a calculus from its theory in this paper.

A λ-theory can be regarded as a category in the usual manner: a type is an object and a function is a morphism. In this sense, the type $\top$ is a terminal object in any λ-theory. However, in a $\lambda\mu$-theory, the type $\bot$ is not an initial object. In this paper, we may use the terminology of the category theory via such translations.

For readability, $A \to \bot$ is denoted by $\neg A$. We also write $A_1 \oplus A_2$ for $\neg\neg A_1 \vee \neg\neg A_2$. As mentioned in Selinger's note [13], $A_1 \oplus A_2$ is more convenient for realistic programs. We use the syntax sugar

$$\iota_j M \equiv \mu(a_1^{\neg\neg A_1}, a_2^{\neg\neg A_2}).\, a_j(\lambda k^{\neg A_j}.\, kM)$$
$$[F_1, F_2] \equiv \lambda x^{B_1 \oplus B_2}.\, \mu a^A.\, (\mu b_2^{\neg\neg B_2}.\, (\mu b_1^{\neg\neg B_1}.\, [b_1, b_2]x)(\lambda x_1^{B_1}.\, a(F_1 x_1)))(\lambda x_2^{B_2}.\, a(F_2 x_2))$$

where all introduced variables are fresh. Then, the typing derivations

$$\frac{\Gamma \vdash M : A_j \mid \Delta}{\Gamma \vdash \iota_j M : A_1 \oplus A_2 \mid \Delta} \qquad \frac{\Gamma \vdash F_1 : B_1 \to A \mid \Delta \quad \Gamma \vdash F_2 : B_2 \to A \mid \Delta}{\Gamma \vdash [F_1, F_2] : B_1 \oplus B_2 \to A \mid \Delta}$$

are admissible. The β-like equality

$$[F_1, F_2](\iota_j M) = F_j M$$

holds but the η-like equality $[\lambda x_1.\, F(\iota_1 x_1), \lambda x_2.\, F(\iota_2 x_2)] = F$ does not hold in general. It follows that $\oplus$ does not give coproducts. When F is a *focal* function as defined below, the η-like equality holds.

Definition 2. A term $F : B \to A$ is focal if $F(\mu a.\, M) = \mu b.\, M\{b(F-)/a-\}$ holds for any term M and a fresh control variable b.

A focal function can be considered an effect-free term-context in the call-by-name $\lambda\mu$-calculus. We define focal term-contexts E as

$$E ::= - \mid EM \mid \lambda x.\, E \mid \pi_1 E \mid \pi_2 E \mid \mu a^A.\, E \mid aE \mid \mu(a^A, a^A).\, E \mid [a, a]E$$

Table 2: Equality of $\lambda\mu$

$$(\lambda x.M)N = M\{N/x\}$$
$$\lambda x.Mx = M \text{ if } x \notin \mathtt{FV}(M)$$
$$\pi_j\langle M_1, M_2\rangle = M_j$$
$$\langle \pi_1 M, \pi_2 M\rangle = M$$
$$\langle\,\rangle = M$$
$$b(\mu a.M) = M\{b/a\}$$
$$\mu a.aM = M \text{ if } a \notin \mathtt{FV}(M)$$
$$[b_1,b_2](\mu(a_1,a_2).M) = M\{b_1,b_2/a_1,a_2\}$$
$$\mu(a_1,a_2).[a_1,a_2]M = M \text{ if } a_1,a_2 \notin \mathtt{FV}(M)$$
$$b^{\perp}M = M$$
$$(\mu a.M)N = \mu b.M\{b(-N)/a-\} \text{ if } b \notin \mathtt{FV}(M)\cup\mathtt{FV}(N)$$
$$\pi_j(\mu a.M) = \mu b.M\{b(\pi_j-)/a-\} \text{ if } b \notin \mathtt{FV}(M)$$
$$[a_1,a_2](\mu a.M) = M\{[a_1,a_2]-/a-\}$$

like evaluation contexts. Note that a focal context is not necessarily an evaluation context because $\lambda x.E$ is not an evaluation context in the usual sense. When free variables of M are not captured in E, we can see that

$$E[\mu a.M] = \mu b.M\{bE/a-\}$$

holds. In later sections, we often use the following fact: for an arbitrary function $F : B \to A$,

$$\neg F \equiv \lambda k^{\neg A}.\lambda x^{B}.k(Fx) \qquad : \neg A \to \neg B$$

is focal.

Categorical characterization of focal functions can be found in [12]. In a $\lambda\mu$-theory, the notion of focal functions coincides with the notion of central functions with respect to $\vee$: a function F is focal if and only if $a(F(\mu b.a'(G(\mu b'.[b,b']x)))) = a'(G(\mu b'.a(F(\mu b.[b,b']x))))$ for any function G.

The notion of focal functions plays an important role in our formulation of inductive types. Focal functions are related to normal functions as follows. For any term $F : B \to A$,

$$F^{\#} \equiv \lambda x^{\neg\neg B}.\mu a^{A}.x(\lambda y^{B}.a(Fy)) \qquad : \neg\neg B \to A$$

is focal. Moreover, this correspondence provides bijection between functions and focal functions. We use the notation

$$F^{\flat} \equiv \lambda x^{B}.F(\lambda k^{\neg B}.kx)$$

for the inverse transformation.

In this paper, a functor means a composition-preserving syntactic transformation on functions of a calculus. We assume a functor has the same codomain as its domain, that is, a functor means an endofunctor on a calculus. In a $\lambda\mu$-theory, we also assume that a functor preserves focal functions. We call a functor defined on focal functions a focus functor. We may call also a focus functor just a functor.

3 Natural Numbers and Streams

Types of natural numbers in programming languages are studied in various ways. A category theoretic approach is one of the most popular characterizations of natural numbers.

In equational extensions of the simply typed λ-calculus without side-effects, the type of natural numbers $\mathbb{N}$ with $\mathtt{zero} : \top \to \mathbb{N}$ and $\mathtt{suc} : \mathbb{N} \to \mathbb{N}$ can be defined as follows: for any $G : \top \to A$ and $F : A \to A$, there exists a unique function $H : \mathbb{N} \to A$ such that each small diagram of

$$\begin{array}{ccccc} \top & \xrightarrow{\mathtt{zero}} & \mathbb{N} & \xleftarrow{\mathtt{suc}} & \mathbb{N} \\ \| & & \downarrow{\scriptstyle H} & & \downarrow{\scriptstyle H} \\ \top & \xrightarrow{G} & A & \xleftarrow{F} & A \end{array}$$

commutes. Such H is denoted by $\mathtt{fold}_{\mathbb{N}}[G,F]$. This natural numbers type can be considered an initial $\mathscr{F}$-algebra, where $\mathscr{F}$ is a functor satisfying $\mathscr{F}A = \top + A$, if a coproduct structure $+$ exists in the calculus.

It is known that a model of the $\lambda\mu$-calculus becomes trivial when it has general initial algebras. Hence, it is not obvious that a natural numbers object can be added to the $\lambda\mu$-calculus consistently.

On the other hand, there exists a consistent model with general final coalgebras. We start from adding a stream type to the $\lambda\mu$-calculus. The type of streams $\mathbb{S}$ with $\mathtt{head}$ and $\mathtt{tail}$ is defined as the dual of natural numbers: for any $G : A \to \bot$ and $F : A \to A$, there exists a unique function $H : A \to \mathbb{S}$ such that each small diagram of

$$\begin{array}{ccccc} \bot & \xleftarrow{\mathtt{head}} & \mathbb{S} & \xrightarrow{\mathtt{tail}} & \mathbb{S} \\ \| & & \uparrow{\scriptstyle H} & & \uparrow{\scriptstyle H} \\ \bot & \xleftarrow{G} & A & \xrightarrow{F} & A \end{array}$$

commutes. We write $\mathtt{unfold}_{\mathbb{S}}\langle G,F\rangle$ for this H. The type of codomain of $\mathtt{head}$ is not $\bot$ in usual programming, but here $\bot$ is enough for simulating natural numbers.

We show that $\neg\mathbb{S}$ behaves like $\mathbb{N}$ in a $\lambda\mu$-theory. We define $\mathtt{zero} : \top \to \neg\mathbb{S}$ and $\mathtt{suc} : \neg\mathbb{S} \to \neg\mathbb{S}$ as

$$\begin{aligned} \mathtt{zero} &\equiv \lambda u^{\top}.\,\mathtt{head} \\ \mathtt{suc} &\equiv \lambda y^{\neg\mathbb{S}}.\lambda x^{\mathbb{S}}.\,y(\mathtt{tail}x) \end{aligned}$$

and write $\overline{n}$ for $\mathtt{suc}^n(\mathtt{zero}\langle\rangle)$. Note that $\overline{n} = \mathtt{head} \circ \mathtt{tail}^n$ holds. Let $H : \neg\mathbb{S} \to A$ be

$$\lambda y^{\neg\mathbb{S}}.\mu a^{A}.y((\mathtt{unfold}_{\mathbb{S}}\langle\lambda h^{\neg A}.h(G\langle\rangle), \lambda k^{\neg A}.\lambda x^{A}.k(Fx)\rangle)(\lambda z.az))$$

for any $G : \top \to A$ and $F : A \to A$. By the equations about $\mathtt{fold}_{\mathbb{S}}$, equations

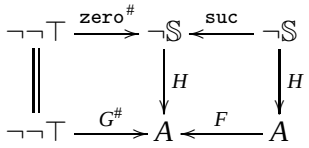
$$\begin{aligned} H\overline{0} &= \mu a.\,\mathtt{head}((\mathtt{unfold}_{\mathbb{S}}\langle\lambda h.h(G\langle\rangle), \lambda k.\lambda x.k(Fx)\rangle)(\lambda z.az)) \\ &= \mu a.\,(\lambda h.h(G\langle\rangle))(\lambda z.az) \\ &= \mu a.\,a(G\langle\rangle) = G\langle\rangle \end{aligned}$$

and

$$\begin{aligned} H(\overline{n+1}) &= \mu a.\overline{n}(\mathtt{tail}((\mathtt{unfold}_{\mathbb{S}}\langle\lambda h.h(G\langle\rangle), \lambda k.\lambda x.k(Fx)\rangle)(\lambda z.az))) \\ &= \mu a.\overline{n}((\mathtt{unfold}_{\mathbb{S}}\langle\lambda h.h(G\langle\rangle), \lambda k.\lambda x.k(Fx)\rangle)((\lambda k.\lambda x.k(Fx))(\lambda z.az))) \\ &= \cdots = F^{n+1}(G\langle\rangle) \end{aligned}$$

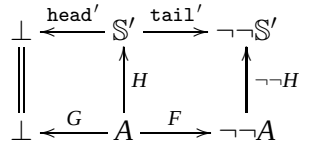
hold. Since control operators are essential for the derivation, such construction is not possible in the pure λ-calculus without side-effects.

Unfortunately, the above $\neg\mathbb{S}$ does not satisfy the property of $\mathbb{N}$ of the λ-calculus because $H(\mathtt{suc}N) = F(HN)$ does not necessarily hold for arbitrary F and N. Instead, if F is focal, $H(\mathtt{suc}N) = F(HN)$ always holds. This fact suggests a possibility that $\neg\mathbb{S}$ may be an initial algebra for focal functions. Since any term of the type $\top \to A$ is never focal, we replace $\top$ with $\neg\neg\top$. One can remember that functions of the type $\top \to A$ and focal functions of $\neg\neg\top \to A$ are in bijective correspondence. Hence, when $F : A \to A$ is focal,



commutes. Note that all functions in the diagram except for F are focal without assumption and $G^{\#}$ can range over all focal functions from $\neg\neg\top$ to A. Because the uniqueness of H in focal functions follows from the uniqueness of $\mathtt{unfold}_{\mathbb{S}}\langle \lambda h.h(G\langle\rangle), \lambda k.\lambda x.k(Fx)\rangle$, the pair of $\mathtt{zero}^{\#}$ and $\mathtt{suc}$ is initial in pairs of focal functions with the above types.

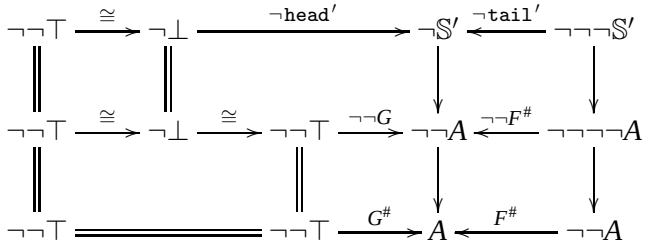
If we use the bijection $-^{\#}$, we can define another type of natural numbers. A modified type of streams $\mathbb{S}'$ is defined as follows: for any $G : A \to \bot$ and $F : A \to \neg\neg A$, the mediating function $H = \mathtt{unfold}_{\mathbb{S}'}\langle G, F\rangle$ satisfies



where $\neg\neg H$ is $\lambda y.\lambda k.y(\lambda x.k(Hx))$. We can define

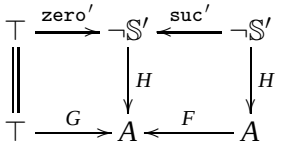
$$\begin{aligned} \mathtt{zero}' &\equiv \lambda u^{\top}.\mathtt{head}' && : \top \to \neg\mathbb{S}' \\ \mathtt{suc}' &\equiv \lambda y^{\neg\mathbb{S}'}.\lambda x^{\mathbb{S}'}.(\mathtt{tail}'x)y && : \neg\mathbb{S}' \to \neg\mathbb{S}' \end{aligned}$$

for a new type of natural numbers $\neg\mathbb{S}'$. For a function $F : B \to A$, let $\neg F : \neg B \to \neg A$ be syntax sugar of $\lambda k.\lambda x.k(Fx)$. For arbitrary functions $G : \top \to A$ and $F : A \to A$, there exist vertical focal functions making each small diagram of



commute. Since $(\mathtt{zero}')^{\#} = \neg\mathtt{head}' \circ \cong$ and $(\mathtt{suc}')^{\#} = \neg\mathtt{tail}'$ hold, we can get a focal function H

such that the diagram



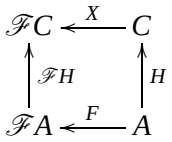
commutes. This H is not unique in general but unique in focal functions. While the equality for $\neg\mathbb{S}$ requires that $F : A \to A$ is focal, the equality for $\neg\mathbb{S}'$ does not.

4 Induction by Coinduction

We generalize the construction of natural numbers from streams shown in the previous section.

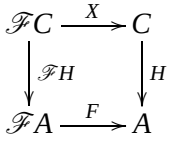
First, we reformulate natural numbers as an initial algebra in focal functions. For fixing the terminology, we give formal definitions about algebras and coalgebras.

Definition 3. Let $\mathscr{F}$ be a functor on a λ-theory. A final $\mathscr{F}$-coalgebra is a type C with a function $X : C \to \mathscr{F}C$ satisfying the condition: for any function $F : A \to \mathscr{F}A$, there is a unique function $H : A \to C$ such that



commutes. We write $\nu\alpha.\mathscr{F}\alpha$ for C and $\mathtt{unfold}F$ for H. An initial $\mathscr{F}$-algebra is the dual of a final $\mathscr{F}$-coalgebra.

Let $\mathscr{F}$ be a focus functor on a $\lambda\mu$-theory. A focally initial $\mathscr{F}$-algebra is a type C with a focal function $X : \mathscr{F}C \to C$ satisfying the condition: for any focal function $F : \mathscr{F}A \to A$, there is a focal function $H : C \to A$ such that



commutes and such H is unique in focal functions. We write $\mu'\alpha.\mathscr{F}\alpha$ for C and $\mathtt{fold}F$ for H.

The symbol μ is used in two meanings for the compatibility of other studies. We can easily classify occurrences of μ: μ in a term means a control operator while μ in a type means a least fixed-point operator.

A focally initial $\mathscr{F}$-algebra means an initial $\mathscr{F}$-algebra in the category of focal functions. Our definition of focally initial algebras is a slightly weaker notion of the definition in Hasegawa's [4]: in that paper, H exists for non-focal F.

Let $\mathscr{F}$ be a functor such that $\mathscr{F}A = \neg\neg\top \vee A$. The function

$$\lambda x^{\neg\neg\top\vee\neg\mathbb{S}}.\mu a^{\neg\mathbb{S}}.a(\mathtt{suc}(\mu b_2.a(\mathtt{zero}^{\#}(\mu b_1.[b_1,b_2]x)))) \qquad : \mathscr{F}(\neg\mathbb{S}) \to \neg\mathbb{S}$$

is a focally initial $\mathscr{F}$-algebra. We can remark that $\vee$ is not a coproduct structure in all functions but is a coproduct structure in focal functions. Since $\top \vee A$ is isomorphic to $\top$ in the $\lambda\mu$-calculus, replacing $\neg\neg\top$ with $\top$ is meaningless unlike the pure λ-calculus.

Let $\mathscr{G}$ be a functor such that $\mathscr{G}A = \bot \times A$. The function

$$\lambda x^{\mathbb{S}}.\langle \mathtt{head}x, \mathtt{tail}x\rangle \qquad : \mathbb{S} \to \mathscr{G}\mathbb{S}$$

is a final $\mathscr{G}$-coalgebra in the usual sense. It is the main theme of this work to investigate when $\neg(\nu\alpha.\mathscr{G}\alpha)$ can be $\mu'\alpha.\mathscr{F}\alpha$.

In order to generalize the construction, we consider the CPS transformation. CPS semantics of the call-by-name $\lambda\mu$-calculus is provided by Selinger [12] based on Hofmann and Streicher's transformation [5]. The target language of the CPS transformation is the simply typed λ-calculus with products and coproducts. Let $[\![-]\!]$ be the transformation on terms and $\overline{-}$ be the transformation on types. A judgment

$$x_1 : B_1, \ldots, x_n : B_n \vdash M : A \mid a_1 : A_1, \ldots, a_m : A_m$$

is transformed to a judgment

$$x_1 : \overline{B_1} \to \mathtt{r}, \ldots, x_n : \overline{B_n} \to \mathtt{r}, a_1 : \overline{A_1}, \ldots, a_m : \overline{A_m} \vdash [\![M]\!] : \overline{A} \to \mathtt{r}$$

for a distinguished type $\mathtt{r}$. We do not show the details of the transformation $[\![-]\!]$ in this paper. The type transformation is given by

$$\begin{aligned}
\overline{\tau} &\equiv \tau \\
\overline{B \to A} &\equiv (\overline{B} \to \mathtt{r}) \times \overline{A} \\
\overline{\top} &\equiv \bot \\
\overline{A_1 \times A_2} &\equiv \overline{A_1} + \overline{A_2} \\
\overline{\bot} &\equiv \top \\
\overline{A_1 \vee A_2} &\equiv \overline{A_1} \times \overline{A_2}
\end{aligned}$$

when we assume the same constant types exist in the target calculus.

In the type transformation, products in the source calculus are translated to coproducts and premonoidal disjunctions are translated to products. It suggests that inductive types should be translated to coinductive types and coinductive types should be translated to inductive types by the CPS transformation. Indeed, $\mathbb{S} = \nu\alpha.\bot \times \alpha$ can be translated to $\mu\alpha.\top + \alpha$, which is the type of natural numbers in the λ-calculus.

In a λ-theory, it is known that $(\mu\alpha.\top + \alpha) \to \mathtt{r}$ is isomorphic to $\nu\alpha.\mathtt{r} \times \alpha$. It means intuitively that a stream is an infinite set of elements indexed by natural numbers. In fact, for the previous $\mathscr{G}$ and $\mathscr{F}$, $\neg(\nu\alpha.\mathscr{G}\alpha) \cong \mu'\alpha.\mathscr{F}\alpha$ in the source calculus is derived from this construction in the target calculus because $\overline{\mathscr{G}A} = \top + \overline{A}$ and $\overline{\mathscr{F}A} \cong \mathtt{r} \times \overline{A}$ hold.

The above construction of a coinductive type from an inductive type in the target calculus can be generalized as follows. If $\mathscr{G}\alpha \to \mathtt{r}$ is naturally isomorphic to $\mathscr{F}(\alpha \to \mathtt{r})$ on α, then

$$(\mu\alpha.\mathscr{G}\alpha) \to \mathtt{r} \cong \nu\alpha.\mathscr{F}\alpha$$

holds. We lift this fact to the source calculus.

We can remark that the following property holds for the CPS transformation: a $\lambda\mu$-term $\lambda x.C[x]$ is focal if there exists a term F such that $[\![C[x]]\!] = \lambda y.x(Fy)$ holds. This property gives a reasoning about focal functions, and is helpful for understanding the following theorem.

Theorem 1. *In a $\lambda\mu$-theory, let $\mathscr{F}$ be a focus functor and $\mathscr{G}$ be a functor such that there exists a natural isomorphism $\neg\mathscr{G}\alpha \cong \mathscr{F}(\neg\alpha)$. For a final $\mathscr{G}$-coalgebra $X : (\nu\alpha.\mathscr{G}\alpha) \to \mathscr{G}(\nu\alpha.\mathscr{G}\alpha)$, if X is focal,*

$$\mathscr{F}(\neg(\nu\alpha.\mathscr{G}\alpha)) \xrightarrow{\cong} \neg\mathscr{G}(\nu\alpha.\mathscr{G}\alpha) \xrightarrow{\neg X} \neg(\nu\alpha.\mathscr{G}\alpha)$$

is a focally initial $\mathscr{F}$-algebra.

The proof of the theorem is as follows. Given a focal function $F : \mathscr{F}A \to A$. Since X is a final $\mathscr{G}$-coalgebra, there exists H such that

$$\begin{array}{ccccccccc}
\mathscr{G}(\nu\alpha.\mathscr{G}\alpha) & & & & \xleftarrow{\quad X\quad} & & & & \nu\alpha.\mathscr{G}\alpha \\
\uparrow{\scriptstyle \mathscr{G}H} & & & & & & & & \uparrow{\scriptstyle H} \\
\mathscr{G}(\neg A) & \xleftarrow{M_{\mathscr{G}(\neg A)}} & \neg\neg\mathscr{G}(\neg A) & \xleftarrow{\cong} & \neg\mathscr{F}(\neg\neg A) & \xleftarrow{\neg\mathscr{F}M_A} & \neg\mathscr{F}A & \xleftarrow{\neg F} & \neg A
\end{array}$$

commutes, where M_A is $\lambda x^{\neg\neg A}.\mu a^A.x(\lambda z.az)$. Let $D : \neg\mathscr{F}A \to \mathscr{G}(\neg A)$ be the function occurring in the bottom-line of the above diagram. Then, the diagram

$$\begin{array}{ccccccc}
\mathscr{F}(\neg(\nu\alpha.\mathscr{G}\alpha)) & \xrightarrow{\cong} & \neg\mathscr{G}(\nu\alpha.\mathscr{G}\alpha) & & \xrightarrow{\quad\neg X\quad} & & \neg(\nu\alpha.\mathscr{G}\alpha) \\
\downarrow{\scriptstyle \mathscr{F}(\neg H)} & & \downarrow{\scriptstyle \neg\mathscr{G}H} & & & & \downarrow{\scriptstyle \neg H} \\
\mathscr{F}(\neg\neg A) & \xrightarrow{\cong} & \neg\mathscr{G}(\neg A) & \xrightarrow{\neg D} & \neg\neg\mathscr{F}A & \xrightarrow{\neg\neg F} & \neg\neg A \\
\downarrow{\scriptstyle \mathscr{F}M_A} & & & & \downarrow{\scriptstyle M_{\mathscr{F}A}} & & \downarrow{\scriptstyle M_A} \\
\mathscr{F}A & = & & = & \mathscr{F}A & \xrightarrow{F} & A
\end{array}$$

commutes. The right upper square is derived from the diagram of the final $\mathscr{G}$-coalgebra. The left upper square comes from the naturality of the isomorphism and the right lower square means that F is focal. Though it is not trivial that the left lower square commutes, the commuting diagram

$$\begin{array}{ccccccc}
\neg\mathscr{G}(\neg A) & \xrightarrow{\neg M_{\mathscr{G}(\neg A)}} & \neg\neg\neg\mathscr{G}(\neg A) & \xrightarrow{\cong} & \neg\neg\mathscr{F}(\neg\neg A) & \xrightarrow{\neg\neg\mathscr{F}M_A} & \neg\neg\mathscr{F}A \\
 & = & \downarrow{\scriptstyle M_{\neg\mathscr{G}(\neg A)}} & & \downarrow{\scriptstyle M_{\neg\neg\mathscr{F}A}} & & \downarrow{\scriptstyle M_{\mathscr{F}A}} \\
 & & \neg\mathscr{G}(\neg A) & \xrightarrow{\cong} & \mathscr{F}(\neg\neg A) & \xrightarrow{\mathscr{F}M_A} & \mathscr{F}A
\end{array}$$

gives a proof. Note that $\mathscr{F}M_A$ is focal because M_A is focal and $\mathscr{F}$ is a focus functor.

For a proof of the uniqueness of the mediating function, we assume that there exists a focal function $H' : \neg(\nu\alpha.\mathscr{G}\alpha) \to A$ satisfying the condition. We can have $H'' \equiv \lambda x.\mu a.x(H'(\lambda z.az)) : \neg A \to (\nu\alpha.\mathscr{G}\alpha)$ such that the diagram for X commutes. The proof of its commutativity requires that X is focal. Hence, $H'' = H$ and $H' = M_A \circ \neg H$ hold.

In the rest of the paper, we show some examples of this construction.

5 Examples

We have already shown that the type of streams $\mathbb{S}$ is the most typical example. Construction of natural numbers from another type of streams $\mathbb{S}'$ is also an example of our result. If we apply the theorem for

$$\mathscr{F}\alpha = \top \oplus \alpha \equiv \neg\neg\top \vee \neg\neg\alpha$$
$$\mathscr{G}\alpha = \neg\top \times \neg\neg\alpha$$

Table 3: Pairs of Functors

$\mathscr{F}\alpha = \neg\neg\alpha$	$\mathscr{G}\alpha = \neg\neg\alpha$
$\mathscr{F}\alpha = \neg B \vee \alpha$	$\mathscr{G}\alpha = B \times \alpha$
$\mathscr{F}\alpha = \alpha \vee \alpha$	$\mathscr{G}\alpha = \alpha \times \alpha$
$\mathscr{F}\alpha = \neg\neg(B \times \alpha)$	$\mathscr{G}\alpha = \neg B \vee \neg\neg\alpha$
$\mathscr{F}\alpha = \neg\neg(\alpha \times \alpha)$	$\mathscr{G}\alpha = \alpha \oplus \alpha$

$\neg\mathbb{S}' \cong \neg(\nu\alpha.\neg\top \times \neg\neg\alpha)$ behaves as $\mu'\alpha.\top \oplus \alpha$. Due to the syntax sugar, $\neg\mathbb{S}'$ is more likely natural numbers type than $\neg\mathbb{S}$. For arbitrary functions G and F, $[G,F]$ is essentially the coproduct arrow of $G^{\#}$ and $F^{\#}$ in the category of focal functions. Hence, in derivations of equations

$$\begin{aligned}(\mathtt{fold}\,[G,F])(\mathtt{zero}'\langle\rangle) &= (\mathtt{fold}\,[G,F])([\mathtt{zero}',\mathtt{suc}'](\iota_1\langle\rangle)) \\ &= [G,F]([\lambda x.\,\iota_1 x, \iota_2 \circ \mathtt{fold}\,[G,F]](\iota_1\langle\rangle)) \\ &= G\langle\rangle\end{aligned}$$

and

$$\begin{aligned}(\mathtt{fold}\,[G,F])(\mathtt{suc}'N) &= (\mathtt{fold}\,[G,F])([\mathtt{zero}',\mathtt{suc}'](\iota_2 N)) \\ &= [G,F]([\lambda x.\,\iota_1 x, \iota_2 \circ \mathtt{fold}\,[G,F]](\iota_2 N)) \\ &= F((\mathtt{fold}\,[G,F])N)\end{aligned}$$

focality conditions are implicit. $\oplus$ is also useful for a list-like data type and a tree-like data type.

Before considering a list-like data type, we consider composition of functors. If we find two pairs of functors satisfying the condition of Theorem 1, also the pair of composition functors satisfies the condition of Theorem 1.

Lemma 2. *In a $\lambda\mu$-theory, if natural isomorphisms $\neg\mathscr{G}_1\alpha \cong \mathscr{F}_1(\neg\alpha)$ and $\neg\mathscr{G}_2\alpha \cong \mathscr{F}_2(\neg\alpha)$ exist, then there exists a natural isomorphism $\neg\mathscr{G}_2\mathscr{G}_1\alpha \cong \mathscr{F}_2\mathscr{F}_1(\neg\alpha)$.*

In Table 3, we show some pairs of primitive functors satisfying the condition of Theorem 1. It is important that the connective $\times$ in the left column is under $\neg\neg$. We do not have a general idea for treating a functor like $B \times -$.

In the λ-calculus without side-effects, a type of B-lists is usually defined as $\mu\alpha.\top + (B \times \alpha)$. Though it is not possible to apply the theorem to $\mu\alpha.\top \vee (B \times \alpha)$ in the $\lambda\mu$-calculus, $\mu\alpha.\top \oplus (B \times \alpha)$ can be used. By the lemma for compositions, the pair of functors

$$\begin{aligned}\mathscr{F}\alpha &= \top \oplus (B \times \alpha) \\ \mathscr{G}\alpha &= \neg\top \times (\neg B \vee \neg\neg\alpha)\end{aligned}$$

satisfies $\neg\mathscr{G}\alpha \cong \mathscr{F}(\neg\alpha)$. Hence, we can get a data type of lists from a coinductive type. If B has a form $\neg C$, the type of B-lists can be expressed by $\neg(\nu\alpha.\neg\top \times (C \oplus \alpha))$.

There is another difficulty caused by the fact that $\vee$ is not monoidal but premonoidal. Since we cannot define a full functor $\mathscr{G}$ so that $\mathscr{G}\alpha = \alpha \vee \alpha$, $\alpha \vee \alpha$ cannot appear in the right column of the table. On the other hand, $\mathscr{F}\alpha = \alpha \vee \alpha$ is permitted in the left column because a focus functor is required to

be defined only on focal functions. While $\alpha \vee \alpha$ is not functorial, it can be seen that $\alpha \oplus \alpha$ is functorial. Therefore, we can define a tree-like data type with a coinductive type. The functors

$$\mathscr{F}\alpha = B \oplus (\alpha \times \alpha)$$
$$\mathscr{G}\alpha = \neg B \times (\alpha \oplus \alpha)$$

induce $\neg(\nu\alpha.\neg B \times (\alpha \oplus \alpha)) \cong \mu'\alpha.B \oplus (\alpha \times \alpha)$. It means that $\neg(\nu\alpha.\neg B \times (\alpha \oplus \alpha))$ behaves like a type of binary trees. Let $\mathbb{S}_{\mathbb{T}}$ be $\nu\alpha.\neg B \times (\alpha \oplus \alpha)$ and

$$\mathtt{head}_{\mathbb{T}} : \mathbb{S}_{\mathbb{T}} \to \neg B$$
$$\mathtt{tail}_{\mathbb{T}} : \mathbb{S}_{\mathbb{T}} \to \mathbb{S}_{\mathbb{T}} \oplus \mathbb{S}_{\mathbb{T}}$$

be functions such that $\lambda x.\langle \mathtt{head}_{\mathbb{T}} x, \mathtt{tail}_{\mathbb{T}} x\rangle$ is a final $\mathscr{G}$-coalgebra. For the type $\mathbb{T} = \neg\mathbb{S}_{\mathbb{T}}$,

$$\mathtt{leaf} \equiv \lambda y^{B}.\lambda x^{\mathbb{S}_{\mathbb{T}}}.(\mathtt{head}_{\mathbb{T}} x)y : B \to \mathbb{T}$$
$$\mathtt{node} \equiv \lambda y^{\mathbb{T}\times\mathbb{T}}.\lambda x^{\mathbb{S}_{\mathbb{T}}}.(\mu a_2.(\mu a_1.[a_1,a_2](\mathtt{tail}_{\mathbb{T}} x))(\pi_1 y))(\pi_2 y) : \mathbb{T}\times\mathbb{T} \to \mathbb{T}$$

can be defined. For any functions $G : B \to A$ and $F : A \times A \to A$, $\mathtt{fold}\,[G,F]$ is defined as

$$\lambda y^{\mathbb{T}}.\mu a^{A}.y((\mathtt{unfold}(\lambda k^{\neg A}.\langle \lambda x^{B}.k(Gx), \mu b^{\neg A \oplus \neg A}.kF'_b\rangle))(\lambda z.az))$$

where F'_b is $F\langle \mu b_1^A.b(\iota_1(\lambda z.b_1 z)), \mu b_2^A.b(\iota_2(\lambda z.b_2 z))\rangle$. Then, we can see that two equations

$$(\mathtt{fold}\,[G,F])(\mathtt{leaf}\,l) = Gl$$
$$(\mathtt{fold}\,[G,F])(\mathtt{node}\,\langle t_1,t_2\rangle) = F\langle(\mathtt{fold}\,[G,F])t_1, (\mathtt{fold}\,[G,F])t_2\rangle$$

hold. Indeed, these equations mean that $\mathbb{T}$ is a data type of binary trees.

6 Concluding Remarks

We have investigated that natural numbers can be expressed by streams in a call-by-name language with control operators. This is a counterpart of the fact that a stream can be expressed by a function from natural numbers in a pure language without side-effects. We have generalized such method and shown construction of inductive types from coinductive types with control operators. As examples of the result, a type of lists and a type of binary trees are expressed by coinductive types.

In our study, while a coinductive type is characterized as a final coalgebra, an inductive type is characterized as an initial algebra in focal functions. In the $\lambda\mu$-calculus, control operators does not permit existence of general initial algebras. We can say that our results give a reasoning about focal restriction of inductive types.

Our work has a possibility to connect the duality between call-by-name and call-by-value [3]. In [12], it is shown that the call-by-value $\lambda\mu$-calculus is the dual of the call-by-name $\lambda\mu$-calculus. Since control operators related to the classical logic, the above duality corresponds to de Morgan's duality. The authors have studied the duality with recursive structures [6, 7], and the second author has provided an extension of the dual calculus with inductive and coinductive types [8]. We are preparing another paper for the duality extended with inductive/coinductive structures and focal restrictions.

Our construction is restricted to some class of functors. Generalization to more complex structure with nested recursion remains to be studied. In the pure λ-calculus, [1] studies a method for more general functors with higher-order functors. It is not obvious to apply this method to the $\lambda\mu$-calculus but may be helpful for generalization of our work.

Acknowledgment

We would like to thank Kazuyuki Asada for discussing the generalization part of this work.

References

[1] T. Altenkirch (2001): *Representation of First Order Function Types as Terminal Coalgebras*. In: *Typed Lambda Calculi and Applications, LNCS* 2044, Springer-Verlag, pp. 8–21, doi:10.1007/3-540-45413-6_5.

[2] G. Barthe & T. Uustalu (2002): *CPS Translating Inductive and Coinductive Types*. In: *Partial Evaluation and Program Manipulation*, ACM Press, pp. 131–142, doi:10.1145/503032.503043.

[3] A. Filinski (1989): *Declarative Continuations and Categorical Duality*. Master's thesis, Computer Science Department, University of Copenhagen.

[4] M. Hasegawa (2006): *Relational Parametricity and Control*. *Logical Methods in Computer Science* 2(3), pp. 1–22, doi:10.2168/LMCS-2(3:3)2006.

[5] M. Hofmann & T. Streicher (1997): *Continuation Models are Universal for $\lambda\mu$-Calculus*. In: *Logic in Computer Science*, IEEE Computer Society, pp. 387–397, doi:10.1109/LICS.1997.614964.

[6] Y. Kakutani (2002): *Duality between Call-by-Name Recursion and Call-by-Value Iteration*. In: *Computer Science Logic, LNCS* 2471, Springer-Verlag, pp. 506–521, doi:10.1007/3-540-45793-3_34.

[7] D. Kimura (2007): *Call-by-Value is Dual to Call-by-Name, Extended*. In: *Asian Symposium on Programming Languages and Systems, LNCS* 4807, Springer-Verlag, pp. 415–430, doi:10.1007/978-3-540-76637-7_28.

[8] D. Kimura & M. Tatsuta (2013): *Call-by-Value and Call-by-Name Dual Calculi with Inductive and Coinductive Types*. *Logical Methods in Computer Science* 9(1:14), pp. 1–38, doi:10.2168/LMCS-9(1:14)2013.

[9] J. Lambek & P. J. Scott (1986): *Introduction to Higher-Order Categorical Logic*. Cambridge University Press.

[10] M. Parigot (1992): *$\lambda\mu$-Calculus: an algorithmic interpretation of classical natural deduction*. In: *Logic Programming and Automated Reasoning, LNCS* 624, Springer-Verlag, pp. 190–201, doi:10.1007/BFb0013061.

[11] G. D. Plotkin (1975): *Call-by-Name, Call-by-Value and the Lambda Calculus*. *Theoretical Computer Science* 1(2), pp. 125–159, doi:10.1016/0304-3975(75)90017-1.

[12] P. Selinger (2001): *Control Categories and Duality: on the categorical semantics of the lambda-mu calculus*. *Mathematical Structures in Computer Science* 11(2), pp. 207–260, doi:10.1017/S096012950000311X.

[13] P. Selinger (2003): *Some Remarks on Control Categories*. Manuscript.