

## A CATEGORICAL MODEL OF $\lambda$ -CALCULUS BASED ON DE BRUIJN'S NAME-FREE EXPRESSIONS

Otsuka, Hiroshi  
Department of Mathematics, Kyushu University

<https://doi.org/10.5109/13400>

---

出版情報 : Bulletin of informatics and cybernetics. 23 (3/4), pp.129-140, 1989-03. Research  
Association of Statistical Sciences

バージョン :

権利関係 :



## A CATEGORICAL MODEL OF $\lambda$ -CALCULUS BASED ON DE BRUIJN'S NAME-FREE EXPRESSIONS

By

**Hiroshi OHTSUKA\***

### Abstract

This paper contains an interpretation of de Bruijn's name-free expression in C-monoid. We show that certain construction of C-monoid (Kleisli C-monoid) corresponds with the expression of  $\lambda$ -calculus by de Bruijn's name-free expression. In particular, the procedure of substitution in de Bruijn's name-free expression, which is more accurate than that in  $\lambda$ -calculus, is systematically characterized in C-monoid.

### 1. Introduction

Just as a cartesian closed category (CCC for short) has been a model of typed  $\lambda$ -calculus, so C-monoid has been a model of  $\lambda$ -calculus. Their relations were pointed out by several authors ([5], [6]). We presented another relation between typed  $\lambda$ -calculus and CCC based on de Bruijn's name-free expressions [3], [4] (abbreviated to DB's NF-exp). Our observation has shown that Kleisli construction (Kleisli category) of CCC characterizes DB's NF-exp and it is isomorphic to polynomial category which is the ordinary model of typed  $\lambda$ -calculus [8].

Terms in DB's NF-exp are the same those of  $\lambda$ -calculus except that they are constructed upon non-negative integers instead of variables. It is important that since DB's NF-exp has no variable, it makes no sense to rename bound variables as was done in  $\lambda$ -calculus using  $\alpha$ -conversion. As a result, the procedure of substitution in DB's NF-exp is defined more explicitly than that in standard notation of  $\lambda$ -calculus.

Now, for the purpose that we want to get the meaning of elaborate substitution in  $\lambda$ -calculus, we adopt DB's NF-exp as a notation and C-monoid as a model. Consequently, we mainly get the following results:

1. Kleisli construction of C-monoid fits the sense of DB's NF-exp.
2. The procedure of substitution in DB's NF-exp is precisely characterized (composition of certain elements) in C-monoid.

Section 2 sketches  $\lambda$ -calculus and DB's NF-exp, in particular, the procedure of substitution in both systems. Section 3 contains a brief introduction of C-monoid and gives a certain construction of it. Section 4 deals with C-monoid as a model of DB's NF-exp (not  $\lambda$ -calculus) and presents the second statement which is one of the main results described above. Considering the relation between C-monoid and CCC, we conclude

---

\* Department of Mathematics, Kyushu University 33, Fukuoka 812, Japan.

that the procedure of substitution is systematically characterized as the composition of certain morphisms in CCC.

## 2. $\lambda$ -calculus and de Bruijn's Name-free Expression

This section presents some notions and notations of  $\lambda$ -calculus and de Bruijn's name-free expression (DB's NF-exp for short), in particular, the procedure of substitution in those systems and relation between them.

The formal system of  $\lambda$ -calculus consists of  $\lambda$ -terms and certain rules between them [1], [6].

DEFINITION 2.1. Let  $x, y, z, \dots$  be infinitely many variables. The set of  $\lambda$ -terms, notation  $A$ , is defined inductively by

1.  $x, y, z, \dots \in A$ ,
2.  $M, N \in A \Rightarrow (MN), (M, N) \in A$ ,
3.  $M \in A$  and a variable  $x \Rightarrow (\lambda x. M), (fst(M)), (snd(M)) \in A$ .

*Free* and *bound* variables are defined as usual.  $FV(M)$  denotes the set of free variables of  $M$ . A term of the form  $(MN)$  is an *application* (of  $M$  to  $N$ ). A term  $(\lambda x. M)$  is a  $\lambda$ -*abstraction*,  $x$  is called *binder* and  $M$  is *body*. A term  $(M, N)$  is a *paring*, a term  $(fst(M))$  is a *1st projection* and  $(snd(M))$  is a *2nd projection*. We adopt usual bracket convention and pairing of variables can be used as bound variable. For example, we permit the following  $\lambda$ -term.

$$(\lambda(x, y).(xy)) \quad (\text{abbreviated to } \lambda(x, y).xy)$$

We use ' $\equiv$ ' to indicate syntactical identity. Next, we define the important operation of *substitution* and rules.

DEFINITION 2.2. It is the replacement of all the free occurrences of a variable in a  $\lambda$ -term with another  $\lambda$ -term. The result of substituting  $N$  for the variable  $x$  in  $M$  is denoted by  $M[x := N]$ . It is defined as follows.

$$x[x := N] \equiv N.$$

$$y[x := N] \equiv y, \quad \text{if } x \not\equiv y.$$

$$(PQ)[x := N] \equiv (P[x := N])(Q[x := N]).$$

$$(P, Q)[x := N] \equiv (P[x := N], Q[x := N]).$$

$$fst(P)[x := N] \equiv fst(P[x := N]).$$

$$snd(P)[x := N] \equiv snd(P[x := N]).$$

$$(\lambda x. P)[x := N] \equiv \lambda x. P.$$

$$(\lambda y. P)[x := N] \equiv \lambda y. (P[x := N]), \quad \text{if } x \not\equiv y \text{ and } y \notin FV(M).$$

$$(\lambda y. P)[x := N] \equiv \lambda z. ((P[y := z])[x := N]),$$

$$\text{if } x \equiv y \text{ and } y \in FV(M), \text{ we choose } z \notin FV(P) \cup FV(N).$$

AXIOM 2.1.  $\lambda$ -terms are to satisfy the following rules.

1.  $\lambda x. M = \lambda y. M[x := y]$ , if  $y \notin FV(M)$ . ( $\alpha$ -conversion)
2.  $(\lambda x. M)N = M[x := N]$ . ( $\beta$ -conversion)
3.  $\lambda x. Mx = M$ , if  $x \notin FV(M)$ . ( $\eta$ -conversion)
4.  $fst(M, N) = M$ ,  $snd(M, N) = N$ ,  
 $(fst(M), snd(M)) = M$ . ( $\pi$ -conversion)
5. standard compatible rules (see Barendregt [1]).

We denote induced equation  $M = N$  by above axioms in  $\lambda$ -calculus for  $\lambda \vdash M = N$ .

The last identity in the definition of substitution may have an ambiguity of selection of bound variable  $z$  ( $\alpha$ -conversion on the body) when we manipulate  $\lambda$ -terms mechanically. To annul it, there is an elegant expression of  $\lambda$ -terms which is called DB's NF-exp together with operations on its terms. It obviates the need for  $\alpha$ -conversion and has more mechanical procedure of substitution than  $\lambda$ -calculus.

The formal system of DB's NF-exp also consists of DB-terms and certain rules between them just as  $\lambda$ -calculus [1], [3], [4].

DEFINITION 2.3. The set of DB-terms, notation  $DB$ , is defined as follows.

1. any non-negative integers are DB-term.
2.  $M, N \in DB \Rightarrow (MN), (M, N) \in DB$ .
3.  $M \in DB \Rightarrow (\lambda. M), (fst(M)), (snd(M)) \in DB$ .

We adopt the same naming and bracket convention as  $\lambda$ -terms and use ' $\equiv$ ' to indicate syntactical identity.

In the standard notation, variables had distinct names, and an occurrence of that variable could be identified by the appearance of its name. We could also distinguish between free and bound occurrence of a variable in a given  $\lambda$ -term. Moreover, given an occurrence of a variable, we could recognize whether it was bound and identify the  $\lambda$  that bound it. An occurrence of  $x$  was bound by the closest surrounding  $\lambda$  that was tagged with the name  $x$ .

DB's NF-exp does away with names for variables and binders for  $\lambda$  while retaining the ability to determine whether and where a particular variable occurrence is bound. Non-negative integer associated with any occurrence  $x$  of a variable, a leaf in the tree representation of  $\lambda$ -term, is the number of nodes labelled  $\lambda y$ , with  $x \neq y$ , which are met in the path from that leaf to the root of this tree (the top level) until a node  $\lambda x$  is encountered. If an integer occurrence  $n$  is greater than the number of  $\lambda$ 's from  $n$  to the top level, then it refers to the free variable.

The definition of substitution in DB's NF-exp is substantially more complicated than the corresponding definition for standard notation. This is mainly because it ensures that no integers of the DB-term to be substituted corresponding with free variables get captured bound variables of the body during substitution. For the purpose of it, we need an appropriate transformation of the DB-term which is substituted.

DEFINITION 2.4. Above transformation is called shift, notation  $U_i^m$ , and is defined as follows.

$$U_i^m(j) \equiv \begin{cases} j & \text{if } j > i \\ j+m & \text{otherwise.} \end{cases}$$

$$\begin{aligned}
U_i^m(MN) &\equiv U_i^m(M)U_i^m(N). \\
U_i^m(\langle M, N \rangle) &\equiv \langle U_i^m(M), U_i^m(N) \rangle. \\
U_i^m(\lambda. M) &\equiv \lambda. U_{i+1}^m(M). \\
U_i^m(fst(M)) &\equiv fst(U_i^m(M)). \\
U_i^m(snd(M)) &\equiv snd(U_i^m(M)).
\end{aligned}$$

We denote the result of substituting  $N$  for the  $m$  in body  $M$  by  $M[m := N]$ .

DEFINITION 2.5. Substitution is defined as follows.

$$\begin{aligned}
n[m := N] &\equiv \begin{cases} n & \text{if } n < m \\ U_0^0(N) & \text{if } n = m \\ n - 1 & \text{if } n > m. \end{cases} \\
(PQ)[m := N] &\equiv (P[m := N])(Q[m := N]). \\
(P, Q)[m := N] &\equiv (P[m := N], Q[m := N]). \\
(\lambda. P)[m := N] &\equiv \lambda. P[m + 1 := N]. \\
fst(P)[m := N] &\equiv fst(P[m := N]). \\
snd(P)[m := N] &\equiv snd(P[m := N]).
\end{aligned}$$

As axioms in DB's NF-exp are the same those of  $\lambda$ -calculus except that  $\alpha$ -conversion is obviated, we only exhibit  $\beta$ ,  $\eta$ ,  $\pi$ -conversion in DB's NF-exp.

AXIOM 2.2. *DB-terms are to satisfy the following axioms and standard compatible rules.*

1.  $(\lambda. M)N = M[0 := N]$ . ( $\beta$ -conversion)
2.  $\lambda. U_0^0(M)0 = M$ . ( $\eta$ -conversion)
3.  $fst(M, N) = M, \text{ snd}(M, N) = N,$  ( $\pi$ -conversion)  
 $(fst(M), snd(M)) = M$ .

We use  $DB \vdash M = N$  to state that  $M = N$  is provable from the axioms in DB's NF-exp.

The next few paragraphs give two translations between  $\lambda$ -calculus and DB's NF-exp which attempt to convince the reader that no essential part of  $\lambda$ -calculus is lost by switching to DB's NF-exp. Free variables are gathered up and make infinite list

$$FV = (x_0, x_1, x_2, \dots).$$

We remark that variables in it are numbered starting with 0.

DEFINITION 2.6. Translation  $\mathfrak{D}: A \rightarrow DB$  under the list  $X$  of free variables is defined as follows.

$$\begin{aligned}
\mathfrak{D}(x, X) &= i, \quad \text{where } i \text{ is the minimum such that } x \equiv x_i \text{ in } X. \\
\mathfrak{D}(MN, X) &= \mathfrak{D}(M, X)\mathfrak{D}(N, X). \\
\mathfrak{D}(\langle M, N \rangle, X) &= (\mathfrak{D}(M, X), \mathfrak{D}(N, X)). \\
\mathfrak{D}(\lambda x. M, X) &= \lambda. \mathfrak{D}(M, (x, X)).
\end{aligned}$$

$$\mathfrak{D}(fst(M), X) = fst(\mathfrak{D}(M, X)).$$

$$\mathfrak{D}(snd(M), X) = snd(\mathfrak{D}(M, X)).$$

The list  $(x, X)$  denotes the result of appending  $X$  to  $(x)$ . At the top level, the list  $X$  is equal with  $FV$ . We remark that the translation on  $\lambda$ -abstraction is performed under the modified list  $(x, X)$  in which certain variable may occur at several positions. Then the variable is translated into the order of the leftmost position of it. For example,

$$\begin{aligned} \mathfrak{D}(\lambda x. xy, (x, y, \dots)) &= \lambda. \mathfrak{D}(x, (x, x, y, \dots)) \mathfrak{D}(y, (x, x, y, \dots)) \\ &= \lambda. 02. \end{aligned}$$

Clearly, if free variables of  $\lambda x. M$  are included in  $X$ , then also are those of  $M$  in  $(x, X)$ .

The inverse translation needs two lists of variables  $FV'$  and  $BV$ . No variable in  $FV'$  appears in  $BV$  and none of  $BV$  also do in  $FV'$ .

**DEFINITION 2.7.** Translation  $\mathfrak{Q}: DB \rightarrow A$  under the lists  $Y$  of free variables and  $Z$  of bound variables is defined as follows.

$$\mathfrak{Q}(i, Y, Z) = x_i, \quad \text{where } x_i \text{ is the } i\text{th variable in } Y.$$

$$\mathfrak{Q}(MN, Y, Z) = \mathfrak{Q}(M, Y, Z) \mathfrak{Q}(N, Y, Z).$$

$$\mathfrak{Q}((M, N), Y, Z) = (\mathfrak{Q}(M, Y, Z), \mathfrak{Q}(N, Y, Z)).$$

$$\mathfrak{Q}(\lambda. M, Y, (x, Z)) = \lambda x. \mathfrak{Q}(M, (x, Y), Z).$$

$$\mathfrak{Q}(fst(M), Y, Z) = fst(\mathfrak{Q}(M, Y, Z)).$$

$$\mathfrak{Q}(snd(M), Y, Z) = snd(\mathfrak{Q}(M, Y, Z)).$$

At the top level, the list  $Y$  is equal with  $FV'$  and  $Z$  is  $BV$ . If  $FV'$  coincides with  $FV$ , we get the following properties which are called syntactic equivalence theorem. Their proofs are straightforward, but tedious computation [8].

**THEOREM 2.1.** Let  $M, N \in A$  and  $P, Q \in DB$ . The following statements hold.

1.  $A \vdash M \equiv_\alpha N \Rightarrow DB \vdash \mathfrak{D}(M, FV) \equiv \mathfrak{D}(N, FV)$ .
2.  $A \vdash M = N \Rightarrow DB \vdash \mathfrak{D}(M, FV) = \mathfrak{D}(N, FV)$ .
3.  $A \vdash M \equiv_\alpha \mathfrak{Q}(\mathfrak{D}(M, FV), FV, BV)$ .
4.  $DB \vdash P = Q \Rightarrow A \vdash \mathfrak{Q}(P, FV, BV) = \mathfrak{Q}(Q, FV, BV)$ .
5.  $DB \vdash P \equiv \mathfrak{Q}(\mathfrak{Q}(M, FV, BV), FV)$ .

Where  $\equiv_\alpha$  denotes the  $\alpha$ -congruent relation.

These relations mention precisely that  $DB$ 's NF-exp has been considered as the  $\alpha$ -congruence (the "real" term) of  $\lambda$ -calculus.

### 3. An Introduction to C-monoid

In this section, we provide a brief description of *C-monoid* and certain construction of it. A C-monoid looks like *cartesian closed category* (CCC for short) except that it lacks the terminal object. It becomes a model of  $\lambda$ -calculus just as CCC does that of typed  $\lambda$ -calculus. Relations between C-monoid and CCC,  $\lambda$ -calculus and typed  $\lambda$ -calculus

are out of the scope of this paper. We do not give their details. Their further references are seen in Koymans [5], Lambek and Scott [6].

DEFINITION 3.1. A C-monoid is a monoid  $\mathfrak{A}$  (identity is  $e$ ) with extra structure  $\langle \pi, \pi', \varepsilon, A, \langle -, - \rangle \rangle$ , where  $\pi, \pi', \varepsilon \in \mathfrak{A}$ ,

$$A: \mathfrak{A} \rightarrow \mathfrak{A},$$

$$\langle -, - \rangle: \mathfrak{A} \times \mathfrak{A} \rightarrow \mathfrak{A},$$

satisfying the following identities:

$$\pi \langle a, b \rangle = a,$$

$$\pi' \langle a, b \rangle = b,$$

$$\langle \pi c, \pi' c \rangle = c,$$

$$\varepsilon \langle A(h)\pi, \pi' \rangle = h,$$

$$A(\varepsilon \langle k\pi, \pi' \rangle) = k,$$

for all  $a, b, c, h$  and  $k \in \mathfrak{A}$ .

First three identities correspond with axiom of *product* in CCC, and so last two ones do axiom of *exponential* in CCC. C-monoids are the objects of a category whose morphisms are C-homomorphisms, which preserve extra structure of C-monoid.

The next few paragraph briefly explains certain construction of C-monoid which is motivated by the corresponding one in a category theory, known as *Kleisli's construction* of certain comonad. Their definitions and concepts are out of scope of the this paper. We do not go into details. Further explanation is referred in MacLane [7]. We only divert Kleisli's method to the construction of C-monoid.

DEFINITION 3.2. Given C-monoid  $\mathfrak{A}$ , Kleisli C-monoid  $\mathfrak{A}_s$  whose elements are the same those of  $\mathfrak{A}$  is defined as follows.

Composition of  $a, b \in \mathfrak{A}_s$  is  $a \circ b = a \langle \pi, b \rangle$ .

Identity of  $\mathfrak{A}_s$  is  $\pi'$ .

Extra structure of  $\mathfrak{A}_s$ ,  $\langle \pi_s, \pi'_s, \varepsilon_s, A_s, \langle -, - \rangle_s \rangle$  is defined as follows.

$$\pi_s = \pi\pi', \quad \pi'_s = \pi'\pi',$$

$$\langle a, b \rangle_s = \langle a, b \rangle,$$

$$\varepsilon_s = \varepsilon\pi',$$

$$A_s(c) = A(c\alpha),$$

where  $a, b, c \in \mathfrak{A}_s$  and  $\alpha = \langle \pi\pi, \langle \pi'\pi, \pi' \rangle \rangle$ .

Above definition indeed gives rise to the structure of C-monoid. We only prove the last two identities in definition 3.1. Let  $h \in \mathfrak{A}_s$ , then first identity is

$$\begin{aligned} \varepsilon_s \circ \langle A_s(h) \circ \pi_s, \pi'_s \rangle_s &= \varepsilon\pi' \langle \pi, \langle A(h\alpha) \langle \pi, \pi\pi' \rangle, \pi'\pi' \rangle \rangle \\ &= \varepsilon \langle A(h\alpha) \langle \pi, \pi\pi' \rangle, \pi'\pi' \rangle \end{aligned}$$

$$\begin{aligned}
&= h\alpha\langle\langle\pi, \pi\pi'\rangle, \pi'\pi'\rangle \\
&= h\langle\pi\pi, \langle\pi'\pi, \pi'\rangle\rangle\langle\langle\pi, \pi\pi'\rangle, \pi'\pi'\rangle \\
&= h\langle\pi, \langle\pi\pi', \pi'\pi'\rangle\rangle \\
&= h.
\end{aligned}$$

Similarly, let  $k \in \mathfrak{A}_s$ , then second one is

$$\begin{aligned}
A_s(\varepsilon_s \circ \langle k \circ \pi_s, \pi'_s \rangle_s) &= A_s(\varepsilon \pi' \langle \pi, \langle k \langle \pi, \pi\pi' \rangle, \pi'\pi' \rangle \rangle \alpha) \\
&= A(\varepsilon \langle k \langle \pi\alpha, \pi\pi'\alpha \rangle, \pi'\pi'\alpha \rangle) \\
&= A(\varepsilon \langle k \langle \pi\pi, \pi'\pi \rangle, \pi' \rangle) \\
&= A(\varepsilon \langle k\pi, \pi' \rangle) \\
&= k.
\end{aligned}$$

□

There is a C-homomorphism  $h : \mathfrak{A} \rightarrow \mathfrak{A}_s$  which sends the element  $a \in \mathfrak{A}$  with  $a\pi' (= a \in \mathfrak{A}_s)$ .

C-monoid and CCC began to be treated as a categorical model of  $\lambda$ -calculus and typed  $\lambda$ -calculus by Koymans [5], Lambek and Scott [6]. But their method dealt with only closed  $\lambda$ -terms. Curien extended their method to be able to treat free variables themselves and adopted so-called polynomial C-monoid as a categorical model of  $\lambda$ -calculus [2].

Polynomial C-monoid, which is usual construction of universal algebra, was introduced by Lambek and Scott [6]. An indeterminate in polynomial C-monoid corresponds with a variable in  $\lambda$ -calculus. On the other hand, it will be turned out that Kleisli's construction of C-monoid corresponds with a non-negative integer in DB's NF-exp.

Next, we translate DB-terms into elements of C-monoid (not polynomial C-monoid) not via  $\lambda$ -calculus but directly. We remark that it does not use the concept of variable and indeterminate.

DEFINITION 3.3. The translation  $\mathfrak{M} : DB \rightarrow \mathfrak{A}$  is defined as follows.

$$\begin{aligned}
\mathfrak{M}(i) &= \pi\pi'^i. \\
\mathfrak{M}(MN) &= \varepsilon \langle \mathfrak{M}(M), \mathfrak{M}(N) \rangle. \\
\mathfrak{M}(\langle M, N \rangle) &= \langle \mathfrak{M}(M), \mathfrak{M}(N) \rangle. \\
\mathfrak{M}(\lambda. M) &= A(\mathfrak{M}(M) \langle \pi', \pi \rangle). \\
\mathfrak{M}(fst(M)) &= \pi(\mathfrak{M}(M)). \\
\mathfrak{M}(snd(M)) &= \pi'(\mathfrak{M}(M)).
\end{aligned}$$

This translation is diverted from the corresponding one between typed  $\lambda$ -calculus and CCC. In it, a non-negative integer is translated to the composition of some projections  $\pi, \pi'$ , which is obtained by successively applying the Kleisli's construction. That is,  $\pi\pi'^i$  is  $i$ th application of  $h$  to  $\pi$ ,  $h(\dots(h(\pi))\dots) = h^i(\pi)$ .

Next section, we will show that the translation  $\mathfrak{M}$  candidates to another categorical model of  $\lambda$ -calculus in the sense of Curien's extention.



#### 4. The Meaning of Substitution in C-monoid

In previous sections, we presented DB's NF-exp which has more accurate procedure of substitution than  $\lambda$ -calculus and obtained new translation which sends DB-terms with elements of C-monoid. For the purpose that we proposes to interpret the accurate procedure of substitution, we adopt DB's NF-exp as a notation and C-monoid as a model. For the rest of this paper,  $\mathfrak{A}$  denotes a C-monoid.

First, we construct the element in C-monoid which corresponds with the operation of shift in DB's NF-exp.

DEFINITION 4.1. Given non-negative integer  $i, m$ , the element  $\gamma_{i, m} \in \mathfrak{A}$  is defined as follows.

$$\gamma_{i, m} = \langle \pi, \gamma_{i-1, m} \pi' \rangle, \quad (i > 0).$$

$$\gamma_{0, m} = \pi'^m, \quad \text{where } \gamma_{0, 0} = e.$$

$\gamma$  has the following simple property,

$$\gamma_{i, m} \gamma_{i, n} = \gamma_{i, m+n}.$$

Shift is interpreted to the right composition of  $\gamma$  by the translation  $\mathfrak{M}$  as follows.

THEOREM 4.1. Let  $M$  be any DB-term and  $i, m$  be non-negative integers. Then, the following equation holds.

$$\mathfrak{M}(U_i^m(M)) = \mathfrak{M}(M) \gamma_{i, m}.$$

Proof is done by induction on the structure of DB-term. We show only two cases, non-negative integer and  $\lambda$ -abstraction.

If  $j < i$ , then,

$$\begin{aligned} \mathfrak{M}(U_i^m(j)) &= \pi \pi'^j \\ \mathfrak{M}(j) \gamma_{i, m} &= \pi \pi'^j \langle \pi, \gamma_{i-1, m} \pi' \rangle \\ &= \pi \pi'^{j-1} \gamma_{i-1, m} \pi' \\ &\quad \vdots \\ &= \pi \gamma_{i-j, m} \pi'^j \\ &= \pi \langle \pi, \gamma_{i-j-1, m} \pi' \rangle \pi'^j \\ &= \pi \pi'^j. \end{aligned}$$

If  $j \geq i$ , then,

$$\begin{aligned} \mathfrak{M}(U_i^m(j)) &= \pi \pi'^{j+m} \\ \mathfrak{M}(j) \gamma_{i, m} &= \pi \pi'^j \langle \pi, \gamma_{i-1, m} \pi' \rangle \\ &= \pi \pi'^{j-1} \gamma_{i-1, m} \pi' \\ &\quad \vdots \\ &= \pi \pi'^{j-i} \gamma_{0, m} \pi'^i \\ &= \pi \pi'^{j-i} \pi'^m \pi'^i \\ &= \pi \pi'^{j+m}. \end{aligned}$$

For  $\lambda$ -abstraction,

$$\begin{aligned}
 \mathfrak{M}(U_i^m(\lambda. M)) &= \mathfrak{M}(\lambda. (U_{i+1}^m(M))) \\
 &= A(\mathfrak{M}(U_{i+1}^m(M)) \langle \pi', \pi \rangle) \\
 &= A(\mathfrak{M}(M) \gamma_{i+1, m} \langle \pi', \pi \rangle) \\
 &= A(\mathfrak{M}(M) \langle \pi', \pi \rangle \langle \gamma_{i, m} \pi, \pi' \rangle) \\
 &= A(\mathfrak{M}(M) \langle \pi', \pi \rangle) \gamma_{i, m} \\
 &= \mathfrak{M}(\lambda. M) \gamma_{i, m}. \quad \square
 \end{aligned}$$

The meaning of substitution needs more complicated element in  $\mathfrak{A}$  which is informally described as performing left rotation.

DEFINITION 4.2. Given non-negative integer  $i, j$  ( $i \geq j$ ), the element  $\delta_{i, j} \in \mathfrak{A}$  is defined as follows.

$$\begin{aligned}
 \delta_{i, j} &= \langle \pi, \delta_{i-1, j-1} \pi' \rangle, \quad (j > 0)! \\
 \delta_{i, 0} &= \langle \pi \pi', \langle \pi \pi'^2, \langle \dots, \langle \pi \pi'^i, \langle \pi, \pi'^{i+1} \rangle \rangle, \dots \rangle \rangle \rangle, \quad \text{where } \delta_{0, 0} = e.
 \end{aligned}$$

If we consider  $\delta_{i, 0}$  as the left rotation of length  $i+1$ ,  $\delta_{0, 0}$  means the left rotation of length 1, which does nothing. Therefore, it is quite all right to decide  $\delta_{0, 0}$  to  $e$ . Now we come to the main result of this paper.

THEOREM 4.2. Let  $M, N$  be DB-terms and  $m$  be non-negative integer. Then the following equation holds.

$$\mathfrak{M}(M[m := N]) = \mathfrak{M}(M) \delta_{m, 0} \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle.$$

We prove only two cases, non-negative integer and  $\lambda$ -abstraction.

If  $n < m$ , then,

$$\begin{aligned}
 \mathfrak{M}(n) \delta_{m, 0} \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle &= \pi \pi'^n \langle \pi \pi', \langle \dots, \langle \pi \pi'^m, \langle \pi, \pi \pi'^{m+1} \rangle \rangle \dots \rangle \rangle \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle \\
 &\vdots \\
 &= \pi \langle \pi \pi'^{n+1}, \langle \dots, \langle \pi \pi'^m, \langle \pi, \pi \pi'^{m+1} \rangle \rangle \dots \rangle \rangle \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle \\
 &= \pi \pi'^{n+1} \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle \\
 &= \pi \pi'^n \\
 &= \mathfrak{M}(n[m := N]).
 \end{aligned}$$

If  $n = m$ , then,

$$\begin{aligned}
 \mathfrak{M}(m) \delta_{m, 0} \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle &= \pi \pi'^m \langle \pi \pi', \langle \dots, \langle \pi \pi'^m, \langle \pi, \pi \pi'^{m+1} \rangle \rangle \dots \rangle \rangle \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle \\
 &\vdots \\
 &= \pi \langle \pi, \pi'^{m+1} \rangle \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle \\
 &= \pi \langle \mathfrak{M}(N) \gamma_{0, m}, e \rangle \\
 &= \mathfrak{M}(N) \gamma_{0, m} \\
 &= \mathfrak{M}(U_0^m(N)) \\
 &= \mathfrak{M}(m[m := N]).
 \end{aligned}$$

If  $n > m$ , then,

$$\begin{aligned}
\mathfrak{M}(n)\delta_{m,0}\langle \mathfrak{M}(N)\gamma_{0,m}, e \rangle &= \pi\pi'^n \langle \pi\pi', \langle \dots, \langle \pi\pi'^m, \langle \pi, \pi\pi'^{m+1} \rangle \rangle \dots \rangle \rangle \langle \mathfrak{M}(N)\gamma_{0,m}, e \rangle \\
&\quad \vdots \\
&= \pi\pi'^{n-m} \langle \pi, \pi'^{m+1} \rangle \langle \mathfrak{M}(N)\gamma_{0,m}, e \rangle \\
&= \pi\pi'^n \langle \mathfrak{M}(N)\gamma_{0,m}, e \rangle \\
&= \pi\pi'^{n-1} \\
&= \mathfrak{M}(n[m := N]).
\end{aligned}$$

For  $\lambda$ -abstraction,

$$\begin{aligned}
\mathfrak{M}(\langle \lambda. M \rangle [m := N]) &= \mathfrak{M}(\lambda. M [m+1 := N]) \\
&= A(\mathfrak{M}(M [m+1 := N]) \langle \pi', \pi \rangle) \\
&= A(\mathfrak{M}(M)\delta_{m+1,0} \langle \mathfrak{M}(N)\gamma_{0,m+1}, e \rangle \langle \pi', \pi \rangle) \\
&= A(\mathfrak{M}(M) \langle \pi\pi', \langle \dots \langle \pi\pi'^{m+1}, \langle \pi, \pi'^{m+2} \rangle \rangle \dots \rangle \rangle \langle \mathfrak{M}(N)\gamma_{0,m}\pi', \langle \pi', \pi \rangle \rangle) \\
&\quad \vdots \\
&= A(\mathfrak{M}(M) \langle \pi', \pi \rangle \langle \langle \pi, \langle \dots, \langle \pi\pi'^{m-1}, \langle \mathfrak{M}(N)\gamma_{0,m}, \pi\pi'^m \rangle \rangle \dots \rangle \rangle \pi, \pi' \rangle) \\
&= A(\mathfrak{M}(M) \langle \pi', \pi \rangle \langle \pi\pi', \langle \dots \langle \pi\pi'^m, \langle \pi, \pi'^{m+1} \rangle \rangle \dots \rangle \rangle \langle \mathfrak{M}(N)\gamma_{0,m}, e \rangle) \\
&= \mathfrak{M}(\lambda. M)\delta_{m,0} \langle \mathfrak{M}(N)\gamma_{0,m}, e \rangle. \quad \square
\end{aligned}$$

Additionally,  $\beta$ -conversion is preserved by the translation  $\mathfrak{M}$ .

COROLLARY 4.1. *Let  $M, N$  be DB-terms. Then,*

$$\mathfrak{M}(\langle \lambda. M \rangle N) = \mathfrak{M}(M[0 := N]).$$

Proof: We put  $m=0$  in above theorem, then,

$$\begin{aligned}
\mathfrak{M}(\langle \lambda. M \rangle N) &= \varepsilon A(\mathfrak{M}(M) \langle \pi', \pi \rangle), \mathfrak{M}(N) \rangle \\
&= \mathfrak{M}(M) \langle \mathfrak{M}(N), e \rangle \\
&= \mathfrak{M}(M)\delta_{0,0} \langle \mathfrak{M}(N)\gamma_{0,0}, e \rangle \\
&= \mathfrak{M}(M[0 := N]). \quad \square
\end{aligned}$$

Moreover the translation  $\mathfrak{M}$  preserves the equality in DB's NF-exp.

THEOREM 4.3. *Let  $M, N$  be DB-terms, then,*

$$DB \vdash M = N \Rightarrow \mathfrak{M}(M) = \mathfrak{M}(N).$$

We prove only the case  $\eta$ -conversion. Let  $M$  be DB-term.

$$\begin{aligned}
\mathfrak{M}(\lambda. U_0^1(M)0) &= A(\mathfrak{M}(U_0^1(M)0) \langle \pi', \pi \rangle) \\
&= A(\varepsilon \langle \mathfrak{M}(M)\gamma_{0,1}, \pi\pi'^0 \rangle \langle \pi', \pi \rangle) \\
&= A(\varepsilon \langle \mathfrak{M}(M)\pi', \pi \rangle \langle \pi', \pi \rangle) \\
&= A(\varepsilon \langle \mathfrak{M}(M)\pi, \pi' \rangle) \\
&= \mathfrak{M}(M). \quad \square
\end{aligned}$$

This theorem mentions that the translation  $\mathfrak{M}$  does not only characterize the procedures of shift and substitution in DB's NF-exp, but also makes C-monoid a model of DB's NF-exp. Consequently, since DB's NF-exp has more explicit procedure of substitution than  $\lambda$ -calculus, we obtain more elaborate model of  $\lambda$ -calculus than ordinary ones.

## 5. Concluding Remarks

This paper is an extended version of [8]. In [8], we got a Kleisli category of CCC as a model of typed  $\lambda$ -calculus and used DB's NF-exp to construct it.

For the purpose that we will interpret DB's NF-exp into a suitable model, we have used C-monoid (type free version of CCC) and have applied the method in [8]. As a result, we can consider the procedure of substitution as the composition of several elements in C-monoid which are systematically constructed. Moreover, we do not use indeterminates which correspond with variables in  $\lambda$ -calculus but elements characterized by non-negative integer in DB's NF-exp.

We appropriated the relation between typed  $\lambda$ -calculus and CCC for that between  $\lambda$ -calculus and C-monoid. In the typed version, there is closed relation between ordinary model (polynomial CCC) and our model (Kleisli category of CCC), which is characterized by the functional completeness of CCC. On the other hand, the relation between polynomial C-monoid and Kleisli C-monoid fades away. The functional completeness of C-monoid does not make those C-monoids isomorphic. This is mainly because C-monoid does not have the element which behaves as the terminal object in CCC. Therefore, our model may not be isomorphic to the ordinary categorical model of  $\lambda$ -calculus.

The studies of categorical investigation of  $\lambda$ -calculus are actively accomplished by Curien [2] et al. They would treat CCC or C-monoid as syntactic systems, e.g., categorical combinatory logic, while we do as description of semantics of  $\lambda$ -calculus. They extend classical combinatory logic to categorical one and try to reconstruct several properties, e.g., syntactic equivalence theorem, confluency under some systems of several axioms.

Because our investigation mainly consider C-monoid as semantics of  $\lambda$ -calculus, it would not be immediately concatenated with their studies. But since we have obtained the categorical meaning of substitution, which is based on DB's NF-exp and therefore quite syntactically characterized, we would like to show some properties of  $\lambda$ -calculus in C-monoid using our result, for instance, sequential substitution or confluency.

## References

- [1] BARENDREGT, H.: *The lambda calculus: syntax and semantics, rev. ed.*, North Holland, Amsterdam, (1984).
- [2] CURIEN, P.-L.: *Categorical combinators*, Inform. Control, **69** (1985), 188-254.
- [3] DE BRUIJN, N.: *Lambda-calculus notation with name-free formulas involving symbols that represent reference transforming mapping*, Indag Math., **40** (1980), 348-356.
- [4] DE BRUIJN, N.: *Lambda-calculus notation with nameless dummies, a tool for automatic formula manipulation*, Indag Math., **34** (1972), 381-392.
- [5] KOYMANS, C. P. J.: *Models of the lambda calculus*, Inform. Control, **52** (1982), 306-332.

- [6] LAMBEK, J. and SCOTT, P.J.: *Introduction to higher order categorical logic*, Cambridge University Press, (1985).
- [7] MACLANE, S.: *Categories for the working mathematician*, Springer-Verlag, New York/Berlin, (1972).
- [8] OHTSUKA, Hiroshi: *Categorical model of typed  $\lambda$ -calculus related to a representation of  $\lambda$ -terms*, Research Report of Mathematics of Computation, Kyushu University, **63-09E** (1988), pp. 29.

*Received September 29, 1988*

*Revised October 17, 1988*

*Communicated by S. Arikawa*