The Theory of Parametricity in Lambda Cube

Takeuti Izumi

Graduate School of Informatics, Kyoto Univ., 606-8501 Japan
takeuti@kuis.kyoto-u.ac.jp

Abstract

This paper is an extension of the work on axiomatisation of the theory parametricity for System F by Plotkin and Abadi into lambda cube. This paper gives the set of axioms of the theory of parametricity for lambda cube. That is done by using conformity relation. Our theory is natural extension of the theory for System F. That is justification of our theory. This paper gives the conjecture that the theory is consistent and the proof is done by relativisation. This paper shows that the theory of parametricity proves the semantic property of dependent direct sum types. This is a merit of our theory.

1 Introduction

1.1 Overview

This paper is an extension of the work on axiomatisation of the theory of parametricity by Plotkin and Abadi [PA] into the system $\lambda P\omega$ in lambda cube, which is studied by Barendregt [Ba]. In their work [PA], Plotkin and Abadi provided an formal logical system for the theory of parametricity over terms of System F, which is second order lambda calculus studied by Girard [G].

In this paper we will give the set of axioms of the theory of parametricity to the system $\lambda P\omega$ in lambda cube. We will define it by using conformity relation, as is similar to the literature [T]. Our theory is natural extension of the theory for System F by Plotkin and Abadi [PA]. That is justification of our theory. Although the consistency of our theory is the definite justification, we have not proved the consistency yet, unfortunately. We have the conjecture that the theory is consistent by the method of relativisation.

As an application, we will show that the theory of parametricity proves the semantic property of dependent direct sum types. This is a merit of our theory.

1.2 Parametricity in Formal Theory

The original meaning of parametricity was as a notion for models of polymorphic calculi, such as System F. More recently, many researchers have had

This is a preliminary version. The final version will be published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

interest in parametricity in formal logic. In this paper we discusses parametricity in formal logic.

We will explain the notion of parametricity in formal logic. Let C be a polymorphic calculation system, and let L be a language of predicate logic over C. Let M be a model of C and $\models$ be the interpretation relation of M. Let $t = t'$ be a candidate of equation in the calculation system C. We write $M \models t = t'$ iff M supports $t = t'$. In other words, the model M supports the calculation $t = t'$ iff the interpretation relation $\models$ holds between M and $t = t'$. It is easy to extend the relation $\models$ into the formulae of L. For each formula P of L, it holds that $M \models_\rho P$ iff the formula P holds in the model M with the environment ρ which assigns free variables in P to suitable objects on M. There may be various models $M, M', M'', \ldots$ with the same interpretation $\models$. There may be also alternative interpretations, such as $\models', \models''$, and so forth. Some of them may be the interpretations for categorical models, and some of them are for set-theoretical models, and so forth. The notion of parametricity in formal logic is formed as a theory T of L such that

$$T = \{\text{ a formula } P \mid M \models P \text{ for any parametric } M\}.$$

The major merit of notion of parametricity in formal logic is to enable us to discuss parametricity independently to various interpretations. After we axiomatise the theory of parametricity, we can derive logical consequences of parametricity from the axioms of parametricity without considering models one by one.

Another merit of formal theory of parametricity is to derive some semantical properties in formal logic. We will discuss this point precisely in the next subsection.

1.3 Semantical Properties

In the studies of informatics, it is important to construct new data types and find out the properties of such data types. For example, let T and T' be data types which is already known. Then we can construct a new data type $T \times T'$ which is the direct product of T and T'. We have functions *left*, *right* and *pair* which satisfy the following equations:

$$\mathit{left}(\mathit{pair}\, xy) = x, \ \ \mathit{right}(\mathit{pair}\, xy) = y \text{ for any } x : T \text{ and } y : T',$$
$$\mathit{pair}(\mathit{left}\, z)(\mathit{right}\, z) = z \text{ for any } z : T \times T'.$$

As for another example, we can construct $\mathit{List}(T)$ which is a type of lists whose components are elements of T. We have the empty list `()` as the elements of $\mathit{List}(T)$, the functions $(-,-)$, which is so called sons-pair, and *listrec*. These element and functions satisfy the following equations:

$$\mathit{listrec}(\,)\,f e = e, \quad \mathit{listrec}(x,l)\,f e = f x(\mathit{listrec}\, l f e)$$

for any $x : T$, $l : \mathit{List}(T)$, $e : D$ and $f : T \to D \to D$, where D is another type. Moreover, we have list induction such that:

Suppose that if $P(l)$ then $P((x,l))$ for any $x : T$ and $l : \mathit{List}(T)$.
Then, if $P(())$ then $P(l)$ for each $l : \mathit{List}(T)$,

where $P(\ )$ is an arbitrary property over $\mathit{List}(T)$.

Polymorphic type theory is very powerful for describing new data types.

One of the most famous polymorphic type theory is System F. Although we have only two type constructor $\to$ and Π in System F, we can encode the direct product $T \times T'$ as $\Pi X.(T \to T' \to X) \to X$ in System F. And also we can encode the functions *left*, *right* and *pair* as some terms in System F. Similarly, the type of the lists is encodes as $\Pi X.(T \to X \to X) \to X \to X$, and the elements and functions `()`, `(−,−)` and *listrec* are also encodable in System F. Under this encoding, we have the following equations up to beta-equivalence:

$$\mathit{left}(\mathit{pair}\,xy) =_\beta x, \quad \mathit{right}(\mathit{pair}\,xy) =_\beta y$$
$$\mathit{listrec}(\,)\,fe =_\beta e, \quad \mathit{listrec}(x,l)\,fe =_\beta fx(\mathit{listrec}\,lfe)$$

But, the term $\mathit{pair}(\mathit{left}\ z)(\mathit{right}\ z)$ is not beta-equivalent to z. Moreover, the statement of list induction is not described in System F, because System F is a system for construction and calculation of terms with type assignment. Thus, we cannot handle the equation $\mathit{pair}(\mathit{left}\ z)(\mathit{right}\ z) = z$ nor list induction with System F. We call such properties semantic properties.

Under such encoding, we have the problem that semantical properties are not derivable. In order to avoid this problem, we sometimes use models of data types which satisfy the semiantical properties, such as the equation $\mathit{pair}(\mathit{left}\ z)(\mathit{right}\ z) = z$ or list induction. Such models are often constructed in category theory or domain theory. But it is more useful to solve that problem only in type theory itself with a formal way.

Our aim is to make formal logical system over terms of a type theory, and to derive semiantical properties by the logical system. A successful example is the formal theory of parametricity over terms of System F, which is studied in some literatures [ACC,PA,T]. The theory of parametricity proves many semantical properties, such as the equation $\mathit{pair}(\mathit{left}\ z)(\mathit{right}\ z) = z$ or list induction. Therefore, we would like to extend the theory of parametricity into more strong or general type theory, namely, the system $\lambda\mathrm{P}\omega$.

1.4 Lambda Cube

Lambda cube is studied by Barendregt [Ba]. It is a family of eight type assignment systems. The system $\lambda 2$ is equivalent to System F, and $\lambda\mathrm{P}\omega$ is equivalent to the system of calculus of construction. We define the theory of parametricity for the system $\lambda\mathrm{P}\omega$ by using conformity relation. It is easy to define the theory of parametricity for each other system of lambda cube in the similar way. We discuss only $\lambda\mathrm{P}\omega$ in this paper because $\lambda\mathrm{P}\omega$ is the most useful in describing data types.

We regard the system $\lambda\mathrm{P}\omega$ as a system of predicate logic over a terms of $\lambda\mathrm{P}\omega$ itself, and define the theory of parametricity for $\lambda\mathrm{P}\omega$ in the system of $\lambda\mathrm{P}\omega$ itself. In his literature [Ba] Barendregt also provides other family of systems named logic cube, in which terms and proofs are separated. But they have more constants and rules than $\lambda\mathrm{P}\omega$. We use $\lambda\mathrm{P}\omega$ because of the simplicity of rules. We can easily separate terms and proofs in the discussion of this paper, and it makes no problem.

1.5 Related Works

In the sense of Reynolds [R], parametricity is the property that if f is of universal type $\Pi X.F[X]$ then $fT(F[R])fU$ holds for all types T and U and for all relations R between the domains of T and of U.

Abadi, Cardelli and Curien [ACC] propose System R, a formal system for parametricity. It is a logical system for binary relations between terms of System F. Because it deals only with binary predicates, its expressive power is rather weak.

Hasegawa [H1,H2,H3] makes a categorical model for polymorphism. If it is parametric, then it is a categorical model for System R. By formalising his informal logic in [H1,H2], we can obtain a formal treatment of parametricity in formal second order predicate logic, Such formal treatment is equivalent to the treatment in [PA,T]. Hasegawa [H1,H2] does not show the existence of the relation-frame which is parametric for all the types of System F, although he shows the existence of the relation-frame which is parametric for some particular types, such as pair types, sum types, natural numbers, and so forth.

Bellucci, Abadi, and Curien [BAC] construct a model of System R from a partial equivalence relation model of System F. This is a model theoretic proof of the consistency of the theory of parametricity.

Plotkin and Abadi [PA] formalise parametricity based on a second order predicate logic. In their system, the basic logic is not specialised to the treatment of parametricity, and the axioms implement the parametricity. Takeuti [T] gives a syntactic proof of the consistency of their system.

As a formal logical systems for dealing with semantic properties, Pfenning and Paulin-Morling [PP] proposed calculus of construction, which is equivalent to $\lambda\mathrm{P}\omega$. They showed that calculus of construction can describe many semiantic properties. The set of axioms which we will give in this paper proves such semiantic properties.

1.6 Outline of This Paper

Section 2 has the brief introduction of conformity relation. Section 3 has the formal definitions of conformity relation and of the axioms of parametricity. Section 4 deals with dependent direct sum types as the application of the theory of parametricity.

2 Conformity Relation

2.1 Overview of the Definition

We will define the axioms of the theory parametricity for $\lambda\mathrm{P}\omega$ The axioms are given as formulae of $\lambda\mathrm{P}\omega$. The conformity relation $\langle - \rangle^{\{\}}$ played the essential role in the theory of axiomatising parametricity for System F in the literature [T]. We would like to define this conformity relation for $\lambda\mathrm{P}\omega$. As for System F, the relation $\langle T \rangle^{\{\Theta\}}$ is defined by induction on T. There are only types which appear as subexpressions of a type. Therefore it is sufficient to define

$\langle T\rangle^{\{\Theta\}}$ only for types T. But in λPω, there are many objects of other levels appears as subexpressions of a type. Therefore we have to define $\langle P\rangle^{\{\Theta\}}$ for P which may not be a type. In order to define that, we define a notation $(\!| P : \alpha |\!)^{\{\Theta\}}$ and call it the kind of conformity relation.

This section explains the outline of conformity relation.

2.2 Conformity Relation for System F

In the work of Plotkin and Abadi [PA], the theory of parametricity is provided by the set of axioms. The axioms is written in second order predicate logic over terms of System F. The formal definition of it appears in the literatures [PA,T]. In order to write the axioms, one defines a binary relation for each type of System F. Here we will call such relations *conformity relations* and write the relation for a type T as $\langle T\rangle$, as is in the literature [T]. The relation $\langle T\rangle$ plays the essential role in defining the set of axioms. The relation $\langle T\rangle$ is defined over the elements of type T. The set of axioms consists of all the formulae of form $\forall x, y : T.\ x = y \supset\subset \langle T\rangle xy$

The relation $\langle T\rangle$ is defined by induction of type T. In order to define $\langle T\rangle$, one defines the auxiliary notation $\langle T\rangle^{\{\Theta\}}$, which denotes the conformity relation with assignment Θ which assigns predicates to type variables. A rough outline of the definition of $\langle T\rangle$ is the following:

$\langle X\rangle^{\{\Theta\}} :\equiv R$, where Θ assigns the predicate R to the type variable X.
$\langle X\rangle^{\{\Theta\}} :\equiv \lambda x^X y^X.\ x = y$,
where Θ assigns no predicate to the type variable X.
$\langle T \to U\rangle^{\{\Theta\}} :\equiv \lambda xy.\forall vw.\langle T\rangle^{\{\Theta\}} vw \supset \langle U\rangle^{\{\Theta\}}(xv)(wy)$.
$\langle \Pi X.T\rangle^{\{\Theta\}} :\equiv \lambda yz.\forall YZ.\forall R^{Y\to Z\to *}.\langle T\rangle^{\{R/X,\Theta\}}(yY)(zZ)$.

$\langle T\rangle :\equiv \langle T\rangle^{\{\}}$, which is with the empty assignment.

In the definition of $\langle \Pi X.T\rangle^{\{\Theta\}}$, one can find the original notion of parametricity. For the type $\Pi X.T \equiv \Pi X.F[X]$, we discussed the relation $F[R]$ for arbitrary types Y and Z and relations R between the domains of Y and of Z. The relation $F[R]$ is represented $\langle F[X]\rangle^{\{R/X\}}$ here. Note that the conformity relation is not a relation between domains of a model but a predicate in a formal logic.

We will also define the theory of parametricity for λPω by the set of axioms. As is similar to the case of System F, we will define conformity relation in λPω, which plays the essential role in defining the set of axioms

2.3 Kinds of Relations

The conformity $\langle T\rangle$ is defined by induction on the type T. System F has only three construction rules of types, which are the rules of type variables, of function types, and of universal types. On the other hand, the system λPω has other rules of construction.

For objects $P : * \to *$ and $Q : *$, we have to define $\langle P\rangle$ before defining the relation $\langle PQ\rangle$. The relation $\langle T\rangle$ is of kind $T \to T \to *$, which is a relation between two elements of the type T. Thus, the kind of $\langle Q\rangle$ is $Q \to Q \to *$ and that of $\langle PQ\rangle$ is $PQ \to PQ \to *$. Therefore, the object $\langle P\rangle$ should work as an object of kind $(Q \to Q \to *) \to PQ \to PQ \to *$. Since Q is independent to P, it seems natural to define $\langle P\rangle$ as an object of kind $\Pi X.(X \to X \to *) \to PX \to PX \to *$, and $\langle PQ\rangle \equiv \langle P\rangle Q\langle Q\rangle$. It is not sufficient, unfortunately. That is because we are defining a binary relation, and different types may be assigned to Q in the both side of the relation. Remember that both types Y and Z are associated to the type variables X in the definition of $\langle \Pi X.T\rangle$ for System F above. Therefore we will define the kind of $\langle P\rangle$ as $\Pi XX'.(X \to X' \to *) \to PX \to PX' \to *$, and $\langle PQ\rangle \equiv \langle P\rangle QQ\langle Q\rangle$.

In general, for $P : \alpha \to *$, one has to know the kind of $\langle Q\rangle$ for $Q : \alpha$ before defining the kind of $\langle P\rangle$. Therefore, for a kind $\alpha : \Box$, we define $(\!|P : \alpha|\!)$ which is the kind of $\langle P\rangle$. In the case $\alpha \equiv *$, it holds that $(\!|P : *|\!) \equiv P \to P \to *$. We call $(\!|P : \alpha|\!)$ *the kind of conformity.*

2.4 Assignments for Variables

In the discussion for System F, one defines $\langle T\rangle^{\{\Theta\}}$, which is the conformity relation with assignment Θ, before defining $\langle T\rangle$. Such assignment is necessary in interpreting type variables. Similarly, we will define conformity relation with the assignment for each system of lambda cube.

In System F, only type variables appear in a type. Thus, it is sufficient to assign predicates only for type variables. On the other hand, in systems of lambda cube, term variables and predicate variables may appear in a type. Therefore we consider assignments which assigns predicates for predicate variables and also terms for term variables.

Remember the case for System F. For a type $X \to Y$, the kind of $\langle X \to Y\rangle^{\{P/X,Q/Y\}}$ is determined from the kinds of P and Q. Suppose P is of $T \to T' \to *$ and Q is of $U \to U' \to *$. Then the predicate $\langle X \to Y\rangle^{\{P/X,Q/Y\}}$ is of $(T \to U) \to (T' \to U') \to *$.

Thus, we assign three objects for a type variable X. The first is the type T with which we substitute X in the type of left argument. The second is the type T' with which we substitute X in the type of right argument. The last is the predicate P.

On the other hand, in defining $\langle P\rangle^{\{\Theta\}}$ where P has a free term variable x, we assign only two objects for x. One is a term with which we substitute x in the type of left argument, and the other is a term with which we substitute x in the type of right argument.

Latter we will define the notions of *double assignment* and *multiple assignment.* A double assignment assigns two objects for predicate variables and term variables. A multiple assignment assigns tree objects for predicate variables and two objects for term variables. We make two substitutions from double assignment or multiple assignment. One of the two substitution makes the type of left argument of a conformity relation, and the other makes the type of right argument.

3 Axioms of Parametricity

3.1 Lambda Cube as a Logical System

Lemma 3.1 *The system* $\lambda P\omega$ *satisfies strong normalisability, confluency and subject reduction property on* β*-reduction.*

Proof. Shown in the literature [Ba]. $\square$

Definition 3.2 (Term, Type, Formula, Predicate, Kind, Basis)
An expression Γ is a *basis* iff $\lambda P\omega$ derives $\Gamma \vdash e : T$. An empty sequence is also a basis.
An expression α is a *kind* iff $\lambda P\omega$ derives $\Gamma \vdash \alpha : \square$.
An expression T is a *type* iff $\lambda P\omega$ derives $\Gamma \vdash T : *$.
A type is also called a *formula*.
An expression P is a *predicate* iff $\lambda P\omega$ derives $\Gamma \vdash P : \alpha$ for some kind α.
An expression t is a *term* iff $\lambda P\omega$ derives $\Gamma \vdash t : T$ for some type T.

Notation 3.2 A notation $A \to B$ is an abbreviation of $\Pi x : A.\, B$, where x is fresh. We write $\lambda x^T.e$ and $\Pi x^T.P$ for $\lambda x{:}T.e$ and $\Pi x{:}T.P$. We write $\lambda xy.e$ for $\lambda x^T.\lambda y^U.e$, and $\Pi xy.P$ for $\Pi x^T.\Pi y^U.P$, and so forth.

Notation 3.3 We write $e =_\lambda f$ for β-equality, because we sometimes use β as a meta-variable.

Notation 3.4 We use the following notations when we regard these types as formulae.

$$
\begin{aligned}
&\forall x^T.P :\equiv \forall x : T.P :\equiv \Pi x : T.\, P,\\
&\exists x^T.P :\equiv \exists x : T.P :\equiv \Pi X : *.(\Pi x : T.P \to X) \to X.\\
&P \supset Q :\equiv P \to Q,\\
&P \wedge Q :\equiv \forall X^*.(P \supset Q \supset X) \supset X,\\
&P \supset\subset Q :\equiv (P \supset Q) \wedge (Q \supset P),\\
&x = y :\equiv x =_T y :\equiv \forall X^{T\to T\to *}.\, Xx \supset Xx,
\end{aligned}
$$

Remark 3.3 We write $\forall$ or $\supset$ instead of Π or $\to$ when we regard it as a logical connective.

Definition 3.4 (Set of Axioms, Theorem, Theory) We regard the set of formulae A as a *set of axioms* only when all the formulae of A is closed, that is, for each $P \in A$, the judgement $\vdash P : *$ is derivable.

Let A be an finite or infinite set of axioms. Let Γ be a basis, which may be empty. We write $A, \Gamma \vdash P$ iff there is a finite sequent $x_1 : A_1, x_2 : A_2, ..., x_3 : A_n$ and a term e such that $A_i \in A$ for all i, and the judgements $\Gamma \vdash P : *$ and $x_1 : A_1, x_2 : A_2, ..., x_n : A_n, \Gamma \vdash e : T$ is derivable.

A formula P is a *theory* of A iff $A \vdash P$. The set of all the theorems of A is called a *theory* of A.

3.5 Kind of Conformity Relation

Definition 3.5 (Double Assignment) A *double assignment* is a set of expressions of the form $(e, f)/x$ which satisfies the followings:

– Each component $(e, f)/x \in \Theta$ consists of a variable x and two objects e and f.
– For each variable x, there is at most one component $(e, f)/x \in \Theta$.

The component $(e, f)/x \in \Theta$ means that Θ assigns the pair of objects (e, f) to the variable x.

Definition 3.6 (Domain of a Double Assignment) Let Θ be a double assignment such that $\Theta = \{(e_1, f_1)/x_1, (e_2, f_2)/x_2, ..., (e_n, f_n)/x_n\}$. Then $dom(\Theta)$ is the set of variable such that $dom(\Theta) = \{x_1, x_2, ..., x_n\}$. It is called the *domain of the double assignment* Θ.

Definition 3.7 (Substitutions of a Double Assignment) Let Θ be a double assignment such that $\Theta = \{(e_1, f_1)/x_1, (e_2, f_2)/x_2, ..., (e_n, f_n)/x_n\}$. Then Θ^l and Θ^r are the substitutions which are defined as

$$\Theta^l = \{e_1/x_1, e_2/x_2, ..., e_n/x_n\} \text{ and } \Theta^r = \{f_1/x_1, f_2/x_2, ..., f_n/x_n\}.$$

Definition 3.8 (Double Assignment for a Basis) Let Γ and Γ' be bases. Let Θ be a double assignment. Then the double assignment Θ is a *double assignment for* Γ *under* Γ' iff Θ satisfies the followings:

– $dom(\Theta) \subset dom(\Gamma)$.
– For each $x : T \in \Gamma$, if $x \notin dom(\Theta)$, then $\Gamma' \vdash x : T$ and $FV(T) \cap dom(\Theta) = \emptyset$.
– For each component $(e, f)/x \in \Theta$, if $\Gamma \vdash x : T$, then $\Gamma' \vdash e : T\Theta^l$ and $\Gamma' \vdash f : T\Theta^r$.

If Θ is a double assignment for Γ under Γ', then we write $\Gamma' \vdash \Gamma \rhd \Theta$.

Definition 3.9 (Kind of Conformity Relation) Let P be a predicate of a kind α and Θ be a double assignment. Then the *kind of conformity relation* of $P : \alpha$ is defined as below:

$$\begin{aligned}
&(\!|P : *|\!)^{\{\Theta\}} :\equiv P\Theta^l \to P\Theta^r \to * \\
&(\!|P : \Pi x^T.\alpha|\!)^{\{\Theta\}} :\equiv \Pi {x_1}^{T\Theta^l} {x_2}^{T\Theta^r}.(\!|Px : \alpha|\!)^{\{\Theta,(x_1,x_2)/x\}} \\
&\qquad \text{for some type } T. \\
&(\!|P : \Pi X^\beta.\alpha|\!)^{\{\Theta\}} :\equiv \\
&\qquad \Pi {X_1}^{\beta\Theta^l} {X_2}^{\beta\Theta^r}.\Pi Y : (\!|X : \beta|\!)^{\{\Theta,(X_1,X_2)/X\}}.(\!|PX : \alpha|\!)^{\{\Theta,(X_1,X_2)/X\}} \\
&\qquad \text{for some kind } \beta.
\end{aligned}$$

Lemma 3.10 *Let Γ and Γ' be bases. Let Θ be a double assignment such that $\Gamma \vdash \Gamma \rhd \Theta$. Let α be a kind and P be a predicate such that $\Gamma \vdash P : \alpha$.*

Then, it is derivable that $\Gamma' \vdash (\!|P : \alpha|\!)^{\{\Theta\}} : \Box$.

Proof. Induction on α. □

Proposition 3.11 *If $\alpha =_\lambda \beta$, then $(\!|P : \alpha|\!)^{\{\Theta\}} =_\lambda (\!|P : \beta|\!)^{\{\Theta\}}$.*

Proof. Easy. □

3.6 Conformity Relation

Definition 3.12 (Multiple Assignment) A *multiple assignment* is a set of expressions of the form $(e, f)/x$ or of the form $(e, f, g)/x$ which satisfies the followings:

- Each component is either of the form $(e, f)/x$ which consists of a variable x and two objects e and f, or of the form $(e, f, g)/x$ which consists of a variable x and three objects e, f and g.
- For each variable x, there is at most one component $(e, f)/x$ or $(e, f, g)/x \in \Theta$.

The component $(e, f)/x \in \Theta$, or $(e, f, g)/x \in \Theta$, means that Θ assigns the pair of objects (e, f), or the triple (e, f, g) respectively, to the variable x.

Definition 3.13 (Domain of a Double Assignment) The domain of a multiple assignment $dom(\Theta)$ is similarly defined to the domain of a double assignment in Definition 3.7.

Definition 3.14 (Double Assignment Made From a Multiple Assignment) For a multiple assignment Θ, the double assignment Θ^{lr} is defined as $(e, f)/x \in \Theta^{lr}$ iff $(e, f)/x \in \Theta$, or $(e, f, g)/x \in \Theta$ for some object g.

In other words, for a multiple assignment Θ, the double assignment Θ^{lr} is made by omitting the third element in each component of Θ with three objects.

Definition 3.15 (Substitutions of a Multiple Assignment) Let Θ be a multiple assignment. Then the substitutions Θ^l and Θ^r are defined as $\Theta^l = (\Theta^{lr})^l$ and $\Theta^l = (\Theta^{lr})^r$.

Definition 3.16 (Multiple Assignment for a Basis) Let Γ and Γ' be bases. Let Θ be a multiple assignment. Then the Multiple assignment Θ is a *multiple assignment for Γ under Γ'* iff Θ satisfies the followings:

- $dom(\Theta) \subset dom(\Gamma)$.
- For each $x : T \in \Gamma$, if $x \notin dom(\Theta)$, then $\Gamma' \vdash x : T$ and $FV(T) \cap dom(\Theta) = \emptyset$.
- For each component $(e, f)/x \in \Theta$, if $\Gamma \vdash x : T$, then the followings are derivable:
$$\begin{array}{ll} \Gamma' \vdash T\Theta^l : *, & \Gamma' \vdash T\Theta^r : *, \\ \Gamma' \vdash e : T\Theta^l, & \Gamma' \vdash f : T\Theta^r. \end{array}$$
- For each component $(e, f, g)/x \in \Theta$, if $\Gamma \vdash x : T$, then the followings are derivable:
$$\begin{array}{ll} \Gamma' \vdash T\Theta^l : \Box, & \Gamma' \vdash T\Theta^r : \Box, \\ \Gamma' \vdash e : T\Theta^l, & \Gamma' \vdash f : T\Theta^r, \\ \text{and} & \Gamma' \vdash g : (\!| x : T |\!)^{\{\Theta\}}. \end{array}$$

If Θ is a double assignment for Γ under Γ', then we write $\Gamma' \vdash \Gamma \rhd \Theta$.

Definition 3.17 (Conformity Relation) Let P be a predicate and Θ be a multiple assignment. Then the *kind of conformity relation* of P is defined as below:

- For a variable X^α,
  $\langle X\rangle^{\{\Theta\}} :\equiv Q \quad$ where $(P_1, P_2, Q)/X \in dom(\Theta)$
  $\langle X\rangle^{\{\Theta\}} :\equiv \ \lambda w_1...w_m xy.$
  $\forall v : (\Pi z_1...z_n.Xz_1...z_n \to *).(vz_1...z_n)\Xi^l x \supset (vz_1...z_n)\Xi^r y$
  where $w_1, ..., w_m$ and $z_1, ..., z_n$ are sequences of variables such that $\alpha \equiv \Pi z_1...z_n.*$, and $(\!|X : \alpha|\!)^{\{\}}$ is expanded into $\Pi w_1...w_m.(\!|Xz_1...z_n : *|\!)^{\{\Xi\}}$ according to Definition 3.9.
- For a type T and a term e,
  $\langle \Pi x : T.P\rangle^{\{\Theta\}} :\equiv \lambda y^{(\Pi x:T.P)\Theta^l} z^{(\Pi x:T.P)\Theta^l}.$
  $\forall {x_1}^{T\Theta^l} {x_2}^{T\Theta^r}.\langle T\rangle^{\{\Theta\}} x_1x_2 \supset \langle T\rangle^{\{\Theta,(x_1,x_2)/x\}}(yx_1)(zx_2)$
  $\langle \lambda x : T.P\rangle^{\{\Theta\}} :\equiv \lambda {x_1}^{T\Theta^l} {x_2}^{T\Theta^r}.\langle Px\rangle^{\{\Theta,(x_1,x_2)/x\}}$
  $\langle Pe\rangle^{\{\Theta\}} :\equiv \langle P\rangle^{\{\Theta\}}(e\Theta^l)(e\Theta^r)$
- For a kind α and a predicate Q,
  $\langle \Pi X : \alpha.P\rangle^{\{\Theta\}} :\equiv$
  $\forall {X_1}^{\alpha\Theta^l} {X_2}^{\alpha\Theta^r}.\forall Y : (\!|X : \alpha|\!)^{\{\Theta^{lr},(X_1,X_2)/X\}}.\langle P\rangle^{\{\Theta,(X_1,X_2,Y)/X\}}$
  $\langle \lambda X : \alpha.P\rangle^{\{\Theta\}} :\equiv$
  $\lambda {X_1}^{\alpha\Theta^l} {X_2}^{\alpha\Theta^r}.\lambda Y : (\!|X : \alpha|\!)^{\{\Theta^{lr},(X_1,X_2)/X\}}.\langle P\rangle^{\{\Theta,(X_1,X_2,Y)/X\}}$
  $\langle PQ\rangle^{\{\Theta\}} :\equiv \langle P\rangle^{\{\Theta\}}(Q\Theta^l)(Q\Theta^r)\langle Q\rangle^{\{\Theta\}}$

Remark 3.18 If Θ assigns noting to a type variable X^*, then $\langle X\rangle^{\{\Theta\}}$ is equivalent to $\lambda x^T y^T.x =_X y$.

Lemma 3.19 *Let Γ and Γ' be bases. Let Θ be a multiple assignment such that $\Gamma \vdash \Gamma \rhd \Theta$ is derivable. Let α be a kind and P be a predicate such that $\Gamma \vdash P : \alpha$ is derivable.*

Then $\Gamma' \vdash \langle P\rangle^{\{\Theta\}} : (\!|P : \alpha|\!)^{\{\Theta^{lr}\}}$ is derivable.

Proof. Induction on P. □

Notation 3.7 If the multiple assignment is empty, then $(\!|P : \alpha|\!) :\equiv (\!|P : \alpha|\!)^{\{\}}$ and $\langle P\rangle :\equiv \langle P\rangle^{\{\}}$. Suppose that $\Gamma \vdash T : *$, $\Gamma \vdash e : T$ and $\Gamma \vdash f : T$. Then $\langle T\rangle ef$ is a formula under the basis Γ. We read this formula $\langle T\rangle ef$ as e is *conforming* to f, or e and f is conforming to each other.

Proposition 3.20 *If $P =_\lambda Q$, then $\langle P\rangle^{\{\Theta\}} =_\lambda \langle Q\rangle^{\{\Theta\}}$.*

Proof. Easy. □

Remark 3.21 (Natural Extension) If P is a type of $\lambda 2$, and all the objects in Θ are in $\lambda 2$, then the conformity relation $\langle P\rangle^{\{\Theta\}}$ is equivalent to that in the theory of axiomatising parametricity for System F. Thus, this definition of conformity is an natural extension of that in the theory of axiomatising parametricity for System F in [PA,T].

Definition 3.22 (Self-conformity, Extensionality) A term t^T is *self-conforming* iff $\langle T\rangle tt$ holds. A type T is *self-conforming* iff $\forall x^T.\langle T\rangle xx$ holds. A predicate $P^{T\to *}$ is extensional iff $\forall x^T y^T.\langle T\rangle xy \supset Px \supset Py$ holds. A type T is *extensional* iff $\forall X^{T\to *}.\forall x^T y^T.\langle T\rangle xy \supset Xx \supset Xy$ holds.

Remark 3.23 The self-conformity of a type T means that for any elements x, y of T, if $x =_T y$ then x is conforming to y. The extensionality of T means the inverse, that is, for any elements x, y of T, if x is conforming to y then $x =_T y$.

Definition 3.24 (Axiom of Parametricity) Let T be a type under basis Γ. Then, *the axiom of parametricity for* T is defined as:

$$\mathbf{Par}(T) :\equiv \forall x^T y^T . x = y \supset\subset \langle T \rangle xy,$$

which is equivalent to the conjunction of self-conformity and extensionality.

The set of all *the axioms of parametricity* is defined as:

$$\mathbf{Par} :\equiv \{\forall \Gamma . \mathbf{Par}(T) \mid \Gamma \vdash T : * \text{ is derivable } \}.$$

3.8 Consistency

Conjecture 3.25 (Consistency) *The set of axioms* **Par** *proves neither* $\forall X^* . X$ *nor* $\forall X^* . \forall x^X y^X . x = y$*, which means corruption of calculation.*

Remark 3.26 We have not proved this conjecture yet, unfortunately. We guess that this conjecture would be proved by the method of relativisation, as is similar to the proof of consistency in [T]. That does not seem easy. We guess that we would have to define manifold relativisation instead of one-fold relativisation which is used in [T]. That is because in $\lambda\mathrm{P}\omega$ the term variables also appear in the type of a term, such as x in e^{Px}. On the other hand, the term variables do not appear in the type of terms in the basic logical system in [PA,T].

4 Application

As an example, we show that the type $\exists x : T . P$ behaves indeed as a type of dependent direct sum in the theory of **Par**.

4.1 Sum Type Dependent on Terms

Proposition 4.1 *Suppose that* $\Gamma \vdash T : *$ *and* $\Gamma, x : T \vdash P : *$*, then*

$$\Gamma \vdash \exists x^T . P : *.$$

Put terms pair, right and left as

$$\begin{aligned} &\mathit{pair} :\equiv \lambda x^T y^P \lambda X^* z^{T \to P \to X} . zxy, \\ &\mathit{left} :\equiv \lambda z^{\exists x : T . P} . zT \lambda x^T y^P . xy^P, \\ &\mathit{right} :\equiv \lambda z^{\exists x : T . P} . z((\lambda x . P)(\mathit{left}\, z)) \lambda x^T y^P . y. \end{aligned}$$

Then we have:

$$\begin{aligned} &\Gamma, x : T, y : P \vdash \mathit{pair}\, xy : \exists x^T . P, \\ &\Gamma, z : \exists x^T : P \vdash \mathit{left}\, z : T, \\ &\Gamma, z : \exists x^T : P \vdash \mathit{right}\, z : (\lambda x . P)(\mathit{left}\, z), \end{aligned}$$

and, moreover,

$$x =_\lambda \mathit{left}(\mathit{pair}\, xy), \ y =_\lambda \mathit{right}(\mathit{pair}\, xy).$$

Theorem 4.2 *Suppose that $\Gamma \vdash T : *$ and $\Gamma, x : T \vdash P : *$, then we have*

$$\mathbf{Par}, \Gamma, z : \exists x^T.P \vdash z = pair(left\ z)(right\ z)$$

Proof. The proof consists of two parts, which are the proof of

$$z = z(\exists x^T.P) pair$$

and the proof of

$$z(\exists x^T.P) pair = pair(left(z(\exists x^T.P)pair))(right(z(\exists x^T.P)pair)).$$

1. First we prove $z = z(\exists x^T.P) pair$.

Suppose

$$\langle \Pi x^T.P \to X\rangle^{\{(X_1,X_2,Y)/X\}} w_1 w_2.$$

For each x_1, x_2, y_1 and y_2,

$$\langle T\rangle x_1 x_2 \supset \langle P\rangle^{\{(x_1,x_2)/x\}} y_1 y_2 \supset Y(w_1 x_1 y_1)(w_2 x_2 y_2).$$

We have

$$w_2 x_2 y_2 =_\lambda pair\, x_2 y_2 X_2 w_2.$$

Put

$$Y' :\equiv \lambda v_1 v_2. Y v_1 (v_2 X_2 w_2).$$

Then,

$$Y(w_1 x_1 y_1)(w_2 x_2 y_2) =_\lambda Y'(w_1 x_1 y_1)(pair\, x_2 y_2).$$

Thus,

$$\langle T\rangle x_1 x_2 \supset \langle P\rangle^{\{(x_1,x_2)/x\}} y_1 y_2 \supset Y'(w_1 x_1 y_1)(pair\, x_2 y_2).$$

Therefore,

$$\langle \Pi x^T.P \to X\rangle^{\{(X_1,\exists x^T.P,Y')/X\}} w_1 pair.$$

Because of self-conformity of $z : \exists x^T.P$, we have

$$Y'(z X_1 w_1)(z(\exists x^T.P) pair),$$

which is expanded into

$$Y(z X_1 w_1)(z(\exists x^T.P) pair\, X_2 w_2),$$

Therefore we have

$$\langle \Pi x^T.P \to X\rangle^{\{(X_1,X_2,Y)/X\}} w_1 w_2 \supset Y(z X_1 w_1)(z(\exists x^T.P) pair\, X_2 w_2).$$

Therefore,

$$\langle \exists x^T.P\rangle z(pair(left\ z)(right\ z)).$$

As the extensionality,

$$z = pair(left\ z)(right\ z).$$

2. Next we prove

$$z(\exists x^T.P) pair = pair(left(z(\exists x^T.P)pair))(right(z(\exists x^T.P)pair)).$$

Because of self-conformity of $z : \exists x^T.P$, we have

$$\langle \Pi x^T.P \to X\rangle^{\{(\exists x^T.P,\exists x^T.P,Y)/X\}} pair\ pair$$
$$\supset Y(z(\exists x^T.P)pair)(z(\exists x^T.P)pair).$$

Suppose

$$\langle T\rangle x_1 x_2 \text{ and } \langle P\rangle^{\{(x_1,x_2/x)\}} y_1 y_2.$$

By the extensionality, we have $x_1 = x_2$ and $y_1 = y_2$. Thus $pair\ x_1 y_1 = pair\, x_2 y_2$.

Hence

$$pair\, x_1 x_2 = pair(left(pair\, x_2 y_2))(right(pair\, x_2 y_2)).$$

Therefore

$$\langle \Pi x^T.P \to X\rangle^{\{(\exists x^T.P, \exists x^T.P, Y)/X\}} pair\ pair$$

where $Y :\equiv \lambda zz'.(z = pair(left\ z')(right\ z'))$

Therefore we have

$$Y(z(\exists x^T.P)pair)(z(\exists x^T.P)pair),$$

which is expanded into

$$z(\exists x^T.P)pair = pair(left(z(\exists x^T.P)pair))(right(z(\exists x^T.P)pair)). \qquad \square$$

4.2 Sum Type Dependent on Predicates

Remark 4.3 Unfortunately, if α is a kind, then we cannot write terms like as *left*, and *right* for a type $\exists X : \alpha.P$. That is because we do not have an object which obtain a predicate out of a term. Nevertheless, we have a term $\mathbf{i}$ like as *pair*, and we can prove the semantic property of $\exists X : \alpha.P$ with this term $\mathbf{i}$.

Notation 4.3 Put a term $\mathbf{i} : \Pi X : \alpha.(P \to \exists X : \alpha.P)$ as:

$$\mathbf{i} \equiv \lambda X^\alpha x^P Y^\alpha y^{\Pi X:\alpha.P\to Y}.yXx.$$

Proposition 4.4 *Put Γ, α, R and P as $\Gamma \vdash \alpha : \square$, $\Gamma \vdash R : \alpha$, and $\Gamma, \vdash M : (\lambda X.P)R$. Then, it holds that $\Gamma \vdash \mathbf{i}RM : \exists X^\alpha.P$*

Proof. Easy. $\square$

Theorem 4.5 *Put Γ, α and P as $\Gamma \vdash \alpha : \square$ and $\Gamma, \vdash P : *$. Then, it holds that*

$$\mathbf{Par}, \Gamma \vdash \forall x^{\exists X:\alpha.P}.\exists X^\alpha y^P.(x = \mathbf{i}Xy).$$

Proof. The proof consists of two parts, which are the proof of

$$x = x(\exists X^\alpha.P)\mathbf{i}$$

and the proof of

$$\exists X^\alpha y^P.(x(\exists X^\alpha.P)\mathbf{i} = \mathbf{i}Xy).$$

1. First we prove $x = x(\exists X^\alpha.P)\mathbf{i}$.

Suppose

$$\langle \Pi X^\alpha.P \to Y\rangle^{\{(Z_1,Z_2,Z_3)/Y\}} z_1 z_2.$$

For each X_1, X_2, X_3, y_1 and y_2 such that

$$X_1 : \alpha,\ X_2 : \alpha,\ X_3 : (\!| X : \alpha |\!),$$
$$y_1 : P\{X_1/X\},\ y_1 : P\{X_2/X\},$$

it holds that

$$\langle P\rangle^{\{(X_1,X_2,X_3)/X\}} y_1 y_2 \supset Z_3(z_1 Z_1 y_1)(z_2 Z_2 y_2).$$

We have

$$z_2 Z_2 y_2 =_\lambda \mathbf{i} Z_2 y_2 X_2 z_2.$$

Put $Q :\equiv \lambda w_1 w_2.Z_3 w_1(w_2 X_2 z_2)$. Then,

$$Z_3(z_1 Z_1 y_1)(z_2 Z_2 y_2) =_\lambda Q(z_1 Z_1 y_1)(\mathbf{i} Z_2 y_2).$$

Thus,

$$\langle P\rangle^{\{(X_1,X_2,X_3)/X\}} y_1 y_2 \supset Q(z_1 Z_1 y_1)(\mathbf{i} Z_2 y_2).$$

Therefore,

$$\langle \Pi X^\alpha.P \to Y\rangle^{\{(Z_1,\exists X.P,Q)/Y\}} z_1 \mathbf{i}.$$

Because of self-conformity of $x : \exists^\alpha X.P$, we have

$$Q(xZ_1w_1)(x(\exists X^\alpha.P)\mathbf{i}),$$

which is expanded into

$$Z_3(xZ_1y_1)(x(\exists X^\alpha.P)\mathbf{i}Z_2y_2),$$

Therefore we have

$$\langle \Pi X^\alpha.P \to Y\rangle^{\{(Z_1,Z_2,Z_3)/Y\}}w_1w_2 \supset Z_3(xZ_1y_1)(x(\exists X^\alpha.P)\mathbf{i}Z_2y_2).$$

Therefore,

$$\langle \exists X^\alpha.P\rangle x(x(\exists X^\alpha.P)\mathbf{i}).$$

As the extensionality,

$$x = x(\exists X^\alpha.P)\mathbf{i}.$$

2. Next we prove $\exists X^\alpha y^P.(x(\exists X^\alpha.P)\mathbf{i} = \mathbf{i}Xy)$.

Put X_1, X_2 and X_3 as:

$$X_1 : P, X_2 : P \text{ and } X_3 : (\!|P|\!),$$

and y_1 and y_2 as:

$$y_1 : (\lambda X.P)X_1,\ y_2 : (\lambda X.P)X_2 \text{ and } \langle P\rangle^{\{(X_1,X_2,X_3)/X\}}y_1y_2.$$

Then it holds that

$$\exists X^\alpha y^P.(\mathbf{i}X_1y_1 = \mathbf{i}Xy).$$

Put R as

$$\lambda z_1^{\exists X.P} z_2^{\exists X.P}.\exists X^\alpha y^P.(z_1 = \mathbf{i}Xy).$$

Then we have $R(\mathbf{i}X_1y_1)(\mathbf{i}X_2y_2)$.

Therefore,

$$\forall X_1X_2X_3.\forall y_1y_2.\langle P\rangle^{\{(X_1,X_2,X_3)/X\}}y_1y_2 \supset R(\mathbf{i}X_1y_1)(\mathbf{i}X_2y_2)$$

which is equivalent to

$$\langle \forall X.P \to Y\rangle^{\{(\exists X.P,\exists X.P,R)/Y\}}\mathbf{ii}$$

Because of self-conformity of $x : \exists X^\alpha.P$, we have

$$\langle(\forall X.P \to Y) \to Y\rangle^{\{(\exists X.P,\exists X.P,R)/Y\}}(x(\exists X.P))(x(\exists X.P))$$

Therefore

$$R(x(\exists X.P)\mathbf{i})(x(\exists X.P)\mathbf{i}),$$

which is equivalent to

$$\exists X^\alpha y^P.(x(\exists X^\alpha.P)\mathbf{i} = \mathbf{i}Xy). \qquad \Box$$

5 Conclusion

We define the theory of parametricity for λPω. That is a natural extension of the theory of parametricity for System F into λPω, as Remark 3.21. This fact justifies our extension. Our theory derives semantical properties of various types. We showed our theory proves the type $\exists x^T.P$ satisfies the property of dependent direct sum.

We have the conjecture that the theory of parametricity is consistent. Unfortunately we have not proved the consistency yet, although the consistency is the definite justification of our theory. We guess that we would prove the consistency by the method of relativisation, as the future work.

Acknowledgement

I am greatful to Sato Masahiko for his support. I am very debt to Kinoshita Yoshiki, Hasegawa Ryu and Hasegawa Masahito for comments, discussions, and encouragements.

References

[ACC] Abadí, M., Cardelli, L., & Curien, P.-L.: Formal parametric polymorphism, *Theoretical Computer Science*, **121**, pp9–58, (1993).

[Ba] Barendregt, H.: Lambda Calculi with Types, *Handbook of Logic in Computer Science,* Vol. 2, pages 117–309, Oxford Univ. Press, 1992, edited by S. Abramsky et al.

[BAC] Bellucci, R., Abadí, M., & Curien, P.-L., A Model for Formal Parametric Polymorphism: A PER Interpretation for system R, *Typed Lambda Calculi and Applications, 1995,* Lecture Notes in Computer Science, 902, pages 32–46 (1995).

[G] Girard J.-Y. et al. *Proofs and Types,* Cambridge Univ. Press, 1989.

[H1] Hasegawa, R.: Parametricity of extensionally collapsed term models of polymorphism and their categorical properties, *Proceedings of Theoretical Aspects of Computer Science,* Lecture Notes In Computer Science 526, pages 495–512, Springer-Verlag, 1991, edited by T. Ito et al.

[H2] Hasegawa, R.: Categorical data types in parametric polymorphism. *Mathematical Structures in Computer Science,* 4:71–109, 1994.

[H3] Hasegawa, R.: Relational limits in general polymorphism. *Publications of the Research Institute for Mathematical Sciences,* Vol. 30 (1994), 535–576.

[PA] Plotkin, G. D., & Abadí, M.: A Logic for parametric polymorphism, *Typed Lambda Calculi and Applications, 1993,* edited by M. Bezem et al., Lecture Notes in Computer Science, 664, pages 361–375, Springer Verlag, 1993.

[PP] Pfenning, F. & Paulin-Mohring, C.: Inductively defined types in the calculus of constructions. *Mathematical foundations of programming semantics* (New Orleans, LA, 1989), 209–228, Lecture Notes in Comput. Sci., 442, Springer, Berlin, 1990.

[R] Reynolds, J. C.: Types, abstraction and parametric polymorphism, R. E. A. Mason, editor, *Information Processing 83,* pages 513–523, Elsevier Science Publishers B. V., 1983.

[T] Takeuti, I.: An Axiomatic System of Parametricity, *Typed Lambda Calculi and Applications, 1997,* Lecture Notes in Computer Science, 1210, pages 354–372 (1997). & *Fundamenta Informatica*, **33**, pp397–432, (1998).