数理解析研究所講究録
第851巻 1993年 139-152

Dinaturality of Simple Subtyping

Yasushi Fujiwara
Systems & Software Engineering Laboratory
Research and Development Center
Toshiba Corporation*

Abstract

Semantic parametricity of a simply-typed λ-calculus with subtyping will be discussed. Viewing subtyping as implicit coercion, we will give a category-theoretic formulation of semantic parametricity. Although semantic parametricity may fail in arbitrary cartesian closed categories (ccc's), simple proof-theoretic argument gives us a sufficient condition on subtyping contexts such that terms satisfy semantic parametricity in arbitrary ccc's.

1 Introduction

Strachey's concept of parametric polymorphism was discussed by Reynolds [24] from the view of representation independence of data types in the study of the second-order λ-calculus. It is expected that the meaning of functions is independent of particular implementations of data types. From then, parametricity of the second-order λ-calculus has been discussed by many researchers [2, 16, 21].

Bounded polymorphism has been discussed in the study of typed object-oriented languages [9]. As the notion of bounded polymorphism is a generalization of the polymorphism in the second-order λ-calculus, it seems natural to expect that the language with bounded polymorphism shows some kind of parametricity. But there have been few discussion on parametricity of bounded polymorphism. (One exception is [7], but its focus is the syntactic equational theory.)

In this article, we will discuss semantic parametricity of a simply-typed λ-calculus with subtyping. Although this system is rather weak compared to Fun or $F_{\leq}$ [10], it serves as a good starting point for the study. Our semantics of subtyping is based on the view "subtyping as implicit coercion." In the calculus with subsumption rule a term may have several types, but coherence [4, 10] guarantees the unicity of the meaning. Thanks to this result, we can formulate the notion of semantic parametricity

*Part of this work was completed while the author was visiting Department of Computer Science, Stanford University.

in a category-theoretic form, dinaturality. In this case, dinaturality implies that functions satisfy "free theorems" [26] that follow from a subtyping context and type information.

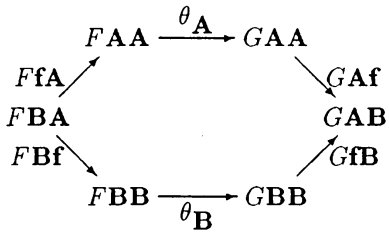
Unfortunately, dinaturality may fail in an arbitrary cartesian closed category (ccc). But we can give a sufficient condition for the subtyping context such that dinaturality holds in arbitrary ccc's. Standard proof-theoretic technique, cut-elimination, plays a crucial role in this result. This extends a result of Girard-Scedrov-Scott [14] that valid typing judgments in a simply-typed λ-calculus provably satisfy dinaturality in arbitrary ccc's.

2 Dinaturality as Semantic Parametricity

In this section, we will review results on parametricity for the calculus *without* subtyping. Roughly speaking, there are two ways of formulating parametricity, logical relation [19] and dinaturality. Reynolds [24] originally formulated it in terms of logical relation. His aim was to establish representation independence theorem (or abstraction theorem) of the the second-order λ-calculus F. Unfortunately the set-theoretic model he discussed was rejected by himself [25], but representation independence of F was proved in [21] by using logical relation of Bruce-Meyer-Mitchell model. Many works in parametricity are based on logical relation. The appealing point of logical relation seems its intuitive relational formulation.

Bainbridge-Freyd-Scedrov-Scott [2] introduced another formulation of semantic parametricity, dinaturality. It had been suggested that the notion of parametric polymorphism was similar to that of natural transformation in category theory viewing types as functors. Obvious trouble of this approach is that type variables may appear in types both positively and negatively so that we may not be able to consider types as (covariant or contravariant) functors. A similar problem was also encountered in category theory and dinatural transformation has been worked out from early sixties [17] as its solution. Although it requires machinery of category theory and, sometimes, proof-theory, dinaturality seems to have something fundamental [3, 14]. The relationship between two formulations had been unclear, but recently Plotkin and Abadi [22] show that under some conditions Reynolds parametricity implies dinaturality. In fact, a result of Girard-Scedrov-Scott [14] also suggests that dinaturality seems to be a primitive form of semantic parametricity. The formulation we adopt in this article will be based on dinatural transformation. The notion of dinaturality is defined equationally as follows:

Definition 1 *Let $F, G : \mathcal{S}^{\circ} \times \mathcal{S} \to \mathcal{T}$ be two functors. Then a family of morphisms $\theta_{\mathbf{A}} : F\mathbf{AA} \to G\mathbf{AA}$ $(\mathbf{A} \in Obj(\mathcal{S}))$ is called a* dinatural transformation *from F to G if the following hexagon diagram commutes for any morphism $f : \mathbf{A} \to \mathbf{B}$ in $\mathcal{S}$.*



$\mathcal{S}^\circ$ denotes the opposite category [17] of $\mathcal{S}$.

Bainbridge-Freyd-Scedrov-Scott [2] is the first to apply dinaturality to the study of parametricity. They showed that in the **PER** model, the interpretation of a typing judgment valid in the second-order λ-calculus F is dinatural. That is, if a typing judgment $x_1 : s_1, \ldots, x_m : s_m \vdash e : t$ is derivable in F and all the free type variables are among $\alpha_1, \ldots, \alpha_n$, then e defines a dinatural transofrmation between two functors $\bar{s}^*, t^* : (\mathbf{PER}^\circ)^n \times \mathbf{PER}^n \to \mathbf{PER}$ corresponding to types $s_1 \times \ldots \times s_m$ and t respectively. The interpretation of the second-order quantifier is given in terms of *end* [17] of dinatural transformations. This may be considered as additional advantage of tis formulation. In this formulation, parametricity is the *definition* of polymorphism, rather than property.

Girard-Scedrov-Scott [14] proved that the interpretation of the simply-typed λ-calculus is dinatural in an *arbitrary* ccc. Let $\mathcal{C}$ be a ccc. If a typing judgment $x_1 : s_1, \ldots, x_m : s_m \vdash e : t$ is derivable in the simply-typed λ-calculus, and all the free type variables are among $\alpha_1, \ldots, \alpha_n$, then e defines a dinatural transofrmation between two functors $\bar{s}^*, t^* : (\mathcal{C}^n)^\circ \times \mathcal{C}^n \to \mathcal{C}$ corresponding to types $s_1 \times \ldots \times s_m$ and t respectively. They used the sequent-calculus version of the calculus and applied the cut-elimination theorem to establish dinaturality. As all rules except cut can be shown to preserve dinaturality, the cut elimination theorem directly implies dinaturality of arbitrary terms.

Thus, dinaturality assures that functions *provably* satisfy equations that are determined only by their types. Several "free theorems" that follow from type information can be found in [14, 26]. In the rest of this article, we will discuss "free theorems" for the calculus with simple subtyping.

3 Dinaturality of Simple Subtyping

In this section, we will formulate dinaturality for a simply-typed λ-calculus *with* subtyping.

3.1 The Calculus $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$

In this subsection, we introduce a simply-typed λ-calculus $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ with subtyping. It can be regarded at the same time as an extension of core-ML language and as a tiny fragment of Fun [9]. In this fragment,

type-variables and subtyping are available, but explicit bounded quantification is prohibited. So, this system is rather weak compared to Fun or $F_{\leq}$. But this gives a good starting point for the study.

We fix the set $\mathcal{L}$ of labels. We will adopt the notational convention that $\alpha, \beta, \ldots$ denote type variables. Then types and raw terms of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ are defined by the following grammar:

$$t ::= \alpha | Top | t \Rightarrow t | \{\ell_1 : t, \ldots, \ell_n : t\}$$

$$e ::= x | \lambda x : t.e | e \cdot e | \{\ell_1 = e, \ldots, \ell_n = e\} | e.\ell$$

We here assume all labels $\ell_1, \ldots, \ell_n$ appearing in the record type $\{\ell_1 : t, \ldots, \ell_n : t\}$ or the record term $\{\ell_1 = e, \ldots, \ell_n = e\}$ are distinct. The type constant *Top* is a supertype of all types.

The subsumption rule of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ makes use of *subtyping judgments* of the form $C \vdash s \leq t$, where C is a *subtyping context.* A subtyping context is a set of *subtyping assertions.* Subtyping assertions are relations of the form $\alpha \leq t$. Subtyping contexts are defined recursively as follows: ϕ is a subtyping context; if C is a subtyping context that does not declare a type variable α and the free type variables of t are already declared in C, then $C, \alpha \leq t = C \cup \{\alpha \leq t\}$ is a subtyping context. Subtyping judgments are defined to be freely generated from the following axioms and formation rules:

$$C \vdash t \leq Top, \text{where the free type variables of } t \text{ are declared in } C \quad (top)$$

$$C_1, \alpha \leq t, C_2 \vdash \alpha \leq t \quad (typevar)$$

$$C \vdash t \leq t, \text{ where the free type variables of } t \text{ are declared in } C \quad (refl)$$

$$\frac{C \vdash s \leq t \quad C \vdash u \leq v}{C \vdash t \Rightarrow u \leq s \Rightarrow v} \quad (\Rightarrow)$$

$$\frac{C \vdash s_{i_1} \leq t_{i_1} \quad \ldots \quad C \vdash s_{i_p} \leq t_{i_p}}{C \vdash \{\ell_1 : s_1, \ldots, \ell_q : s_q\} \leq \{\ell_{i_1} : t_{i_1}, \ldots, \ell_{i_p} : t_{i_p}\}} \quad (recd)$$

$$\frac{C \vdash r \leq s \quad C \vdash s \leq t}{C \vdash r \leq t} \quad (trans)$$

In (*recd*), we assume that $1 \leq i_1 < i_2 < \ldots < i_p \leq q$.

Raw terms are type-checked by deriving *typing judgments* of the form $C, A \vdash e : t$, where C is a subtyping context and A is a *typing context.* Typing contexts are a set of relations of the form $x : t$, with no variable x occuring twice. In addition to ordinary formation rule for typing judgments like ($\Rightarrow$, *intro*),($\Rightarrow$,*elim*), (*recd, intro*), and (*recd, elim*), $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ has the following subsumption rule.

$$\frac{C, A \vdash e : s \quad C \vdash s \leq t}{C, A \vdash e : t} \quad (subsumption)$$

Given our fourmulation of terms, it is natural to write equations of the form $C, A \vdash e = e' : t$, where we assume that both $C, A \vdash e : t$ and $C, A \vdash e' : t$ hold. The equality is a congruence relation defined by ordinary rules like α-, β-,η-, *recd*-β-, *recd*-η-rules.

3.2 Coercion Semantics and Dinaturality

In the rest of this article, we will try to extend the result of Girard-Scedrov-Scott to $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$. In this subsection we will discuss the setting for the meaning of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$terms using category theory. The idea essential in the following is to consider not only types but also subtyping relations. As we allow subtyping contexts like $C = \{\alpha \leq Top, \beta \leq \alpha \Rightarrow \alpha\}$, the morphisms between objects should have some relatioships to these subtyping assertions. This is clarified by the view that "subtyping as implicit coercion." [4, 10, 24] Emulating subsumption rule by explicit coercions, we will give the meanings of terms as families of morphisms in cartesian closed categories.

It is rather well-known that the theory of typed λ-calculus is equivalent to that of cartesian closed categories (ccc's). We here give a brief review of ccc's. For further information, the reader is referred to [1, 15, 19]. As ccc's can be viewed as a slight generalization of well-known Henkin models [19], there will be little fear of confusion if readers unfamiliar to category theory take ccc's for Henkin models.

A *cartesian closed category* is a category with a specified terminal object $\mathbf{1}$, products and exponentials. This means that each ccc has the following specific data:

- object $\mathbf{1}$ with a unique morphism $\mathcal{O}^s : s \to \mathbf{1}$ for each object s.
- binary object map $\times$ with, for any objects s, t and u, specified morphisms $\pi^1_{s,t} : s \times t \to s, \pi^2_{s,t} : s \times t \to t$, and a map $\langle , \rangle : Mor(u,s) \times Mor(u,t) \to Mor(u, s \times t)$ such that, for every $f : u \to s$ and $g : u \to t$, the morphism $\langle f, g \rangle : u \to s \times t$ is the unique h satisfying $f = h; \pi^1_{s,t}$ and $g = h; \pi^2_{s,t}$.
- binary object map $\Rightarrow$ with, for any objects s, t and u, a specified morphism $App : (s \Rightarrow t) \times s \to t$ and a map $Curry : Mor(s \times t, u) \to Mor(s, t \Rightarrow u)$ such that for every $f : s \times t \to u, Curry(f) : s \to (t \Rightarrow u)$ is the unique h satisfying $\langle \pi^1_{s,t}; h, \pi^2_{s,t} \rangle; App = f$.

Here we denote the composition of two morphisms $f : s \to t$ and $g : t \to u$ by $f; g : s \to u$. We will let $\times$ associate to the right.

Let $\mathcal{C}$ be a ccc. For any type t of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$, we will define its interpretation as a functor $t^* : (\mathcal{C}^\circ)^n \times \mathcal{C}^n \to \mathcal{C}$, where $\alpha_1, \ldots, \alpha_n$ are all type variables occuring in t. For $\mathbf{a} = (a_1, \ldots, a_n), \mathbf{b} = (b_1, \ldots, b_n) \in (Obj(\mathcal{C}))^n$, let us define the object $t^*\mathbf{ab}$ inductively as follows:

$$
\begin{aligned}
(\alpha_i)^*\mathbf{ab} &= b_i \\
(Top)^*\mathbf{ab} &= \mathbf{1} \\
(s \Rightarrow t)^*\mathbf{ab} &= s^*\mathbf{ba} \Rightarrow t^*\mathbf{ab} \\
\{\ell_1 : t_1, \ldots, \ell_n : t_n\}^*\mathbf{ab} &= t_1^*\mathbf{ab} \times \ldots \times t_n^*\mathbf{ab}
\end{aligned}
$$

In order to consider the typing judgments, we first need to define coercions, which will be used to emulate subsumption rule. Let $\alpha_1, \ldots, \alpha_n$ be type variables that occur in a subtyping context C. A pair

$(\mathbf{a}, \mathbf{f})$ consisting of a tuple of objects $\mathbf{a} = (a_1, \ldots, a_n) \in (Obj(\mathcal{C}))^n$ and that of morphisms $\mathbf{f} = (f_1, \ldots, f_n)$ with $f_i : a_i \to t_i^*\mathbf{aa}$ for $\alpha_i \leq t_i \in C$ will be called a *coercion pair* and a morphism f_i will be called a *coercion morphism* corresponding to a subtyping assertion $\alpha_i \leq t_i$.

We will give the meaning of typing judgments $C, A \vdash e : t$ as a family of morphisms $[\![e]\!]_{\mathbf{a},\mathbf{f}} : s_1^*\mathbf{aa} \times \ldots \times s_n^*\mathbf{aa} \to t^*\mathbf{aa}$ for $(\mathbf{a}, \mathbf{f})$, where $A = \{x_i : s_i | 1 \leq i \leq m\}$. We will abuse notation and write $\bar{s}^*\mathbf{aa}$ for $s_1^*\mathbf{aa} \times \ldots \times s_n^*\mathbf{aa}$. We have to notice, however, that it is possible that a typing judgment corresponds to several different morphisms in $\mathcal{C}$ because of subsumption rule. This problem is nontrivial in general and called *coherence* [4]. In this case, we can guarantee that the meaning of a typing judgment is unique thanks to [4].

In our formulation, families of morphisms are also attached to subtyping judgments. As a subtyping judgment also may have multiple derivations, we first attach a family of morphisms to a derivation of a subtyping judgment. We associate a family of morphisms $[\![\sigma]\!]_{\mathbf{a},\mathbf{f}} : s^*\mathbf{aa} \to t^*\mathbf{aa}$ to a derivation σ of a subtyping judgment $C \vdash s \leq t$. For the explicit construction of morphisms, readers should consult, for example, [4].

We will associate a family of morphisms $[\![\Delta]\!]_{\mathbf{a},\mathbf{f}} : \bar{s}^*\mathbf{aa} \to t^*\mathbf{aa}$ to a derivation Δ of a typing judgment $C, A \vdash e : t$ with $A = \{x_i : s_i | 1 \leq i \leq m\}$. The procedure is almost identical to the one discussed in [19]. The main difference lies, of course, in the subsumption rule, but the necessary modification is obvious. If Δ is a derivation of the judgment $C, A \vdash e : t$ from a derivation Δ_1 of $C, A \vdash e : s$ and a derivation σ of a subtyping judgment $C \vdash s \leq t$ by (*subsumption*), then we define

$$[\![\Delta]\!]_{\mathbf{a},\mathbf{f}} = [\![\Delta_1]\!]_{\mathbf{a},\mathbf{f}}; [\![\sigma]\!]_{\mathbf{a},\mathbf{f}} : \bar{s}^*\mathbf{aa} \to t^*\mathbf{aa}$$

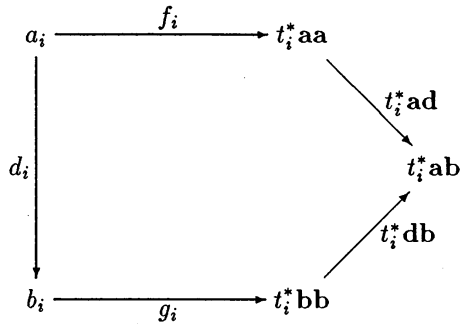
Proposition 1 *1. Suppose that σ_1 and σ_2 be two derivations of a subtyping judgment $C \vdash s \leq t$. Then we have $[\![\sigma_1]\!]_{\mathbf{a},\mathbf{f}} = [\![\sigma_2]\!]_{\mathbf{a},\mathbf{f}} : s^*\mathbf{aa} \to t^*\mathbf{aa}$ for any coercion pair $(\mathbf{a}, \mathbf{f})$.*

2. Suppose that Δ_1 and Δ_2 be two derivations of a typing judgment $C, A \vdash e : t$. Then we have $[\![\Delta_1]\!]_{\mathbf{a},\mathbf{f}} = [\![\Delta_2]\!]_{\mathbf{a},\mathbf{f}}$ for any coercion pair $(\mathbf{a}, \mathbf{f})$.

Thus, the meaning of a judgment is uniquely defined. We may denote the meaning of $C \vdash s \leq t$ by $[\![s \leq t]\!]_{\mathbf{a},\mathbf{f}} : s^*\mathbf{aa} \to t^*\mathbf{aa}$ and that of $C, A \vdash e : t$ by $[\![e]\!]_{\mathbf{a},\mathbf{f}} : \bar{s}^*\mathbf{aa} \to t^*\mathbf{aa}$.

Now we can give our formulation of dinaturality. As we have noticed, we have to take consideration of not only type instances but also coercion pairs. Intuitively speaking, subtyping assertions are similar to proper axioms. (This claim will be made clear in the course of proof-theoretic analysis below.) So, it is natural to assume that coercion morphisms are also dinatural with respect to morphisms between type instances. It in turn means that we should only consider morphims between coercion pairs such that coercion morphisms are dinatural with respect to them. Notice that similar modification of Reynold's parametricity was necessary when the fixpoint operator was added to the pure λ-calculus [26].

Let $(\mathbf{a}, \mathbf{f})$ and $(\mathbf{b}, \mathbf{g})$ be two coercion pairs with respect to the subtyping context $C = \{\alpha_i \leq t_i | 1 \leq i \leq n\}$. A tuple $\mathbf{d} = (d_1, \ldots, d_n)$ with $d_i \in Mor(a_i, b_i)$ will be called a morphism from $(\mathbf{a}, \mathbf{f})$ to $(\mathbf{b}, \mathbf{g})$ if it makes the following pentagon diagram commutative for $1 \leq i \leq n$.



In fact, this terminology is legal, that is, coercion pairs form a category with respect to morphisms we just defined.

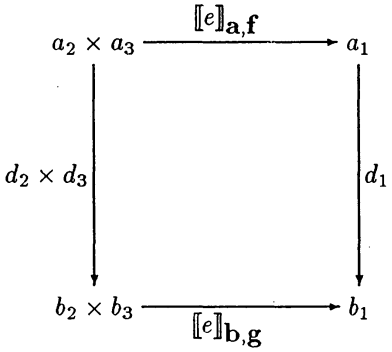
Lemma 1 *Morphisms between coercion pairs compose.*

Let us denote by $\mathcal{C}_C$ the category defined above. In the case that subtyping assertions are all of the form $\alpha_i \leq t_i$ with t_i constant types, the above condition reduces to $f_i = d_i; g_i$. But in the case that type variables appear in t_i, the situation is necessarily a little more complicated.

By considering the forgetful functor from $\mathcal{C}_C$ to $\mathcal{C}^n$ given by $(\mathbf{a}, \mathbf{f}) \rightarrow \mathbf{a}$, we can associate a type t with a functor $t^* : (\mathcal{C}_C)^\circ \times \mathcal{C}_C \rightarrow \mathcal{C}$. Notice that, if the subtyping context is of the form $C = \{\alpha_i \leq Top | 1 \leq i \leq n\}$, then $\mathcal{C}_C$ is $\mathcal{C}^n$. Semantic parametricity we expect is formalized as follows:

Claim Let $\mathcal{C}$ be a ccc and suppose that $C, A \vdash e : t$ is provable in $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$. Then the family $[\![e]\!]_{\mathbf{a},\mathbf{f}}$ is a dinatural transformation between two functors $\bar{s}^*, t^* : (\mathcal{C}_C)^\circ \times \mathcal{C}_C \rightarrow \mathcal{C}$, where $A = \{x_i : s_i | 1 \leq i \leq m\}$.

Example Let us discuss a simplifed version of an example given in [10]. Let $C = \{\alpha_1 \leq Top,\ \alpha_2 \leq \alpha_1,\ \alpha_3 \leq \alpha_1\}$ be a subtyping context. Let $(\mathbf{a}, \mathbf{f})$ and $(\mathbf{b}, \mathbf{g})$ be objects of $\mathcal{C}_C$. Then, $f_1 = \mathcal{O}^{a_1} : a_1 \rightarrow \mathbf{1}, f_2 : a_2 \rightarrow a_1$, and $f_3 : a_3 \rightarrow a_1$. By definition, the morphism $\mathbf{d} : (\mathbf{a}, \mathbf{f}) \rightarrow (\mathbf{b}, \mathbf{g})$ satisfies the equations $f_2; d_1 = d_2; g_2 : a_2 \rightarrow b_1$ and $f_2; d_1 = d_2; g_2 : a_2 \rightarrow b_1$. Then the above claim implies that if a typing judgment $C, x : \alpha_2, y : \alpha_3 \vdash e : \alpha_1$ is valid, then the corresponding morphisms $[\![e]\!]_{\mathbf{a},\mathbf{f}}$ and $[\![e]\!]_{\mathbf{b},\mathbf{g}}$ make the following diagram commutative.



Typing judgments $C, x : \alpha_2, y : \alpha_3 \vdash x : \alpha_1$ and $C, x : \alpha_2, y : \alpha_3 \vdash y : \alpha_1$ are valid. It is easy to see our claim holds for these judgments. For example, as the morphism $[\![x]\!]_{\mathbf{a,f}}$ corresponding to the judgment $C, x : \alpha_2, y : \alpha_3 \vdash x : \alpha_1$ is given by $\pi^1_{a_2,a_3}; f_2 : a_2 \times a_3 \to a_1$, we easily see that the above diagram is commutative. □

Unfortunately, this claim is *not* true for an arbitrary ccc. Let us consider a subtyping context $C = \{\alpha \leq Top, \beta \leq (\alpha \Rightarrow \alpha) \Rightarrow \alpha\}$. Suppose that there exists a dinatural transformation $Y : (\alpha \Rightarrow \alpha) \to \alpha$ with respect to a ccc $\mathcal{C}$. By considering the currification, we get a family of morphisms $Curry(Y)_a \in Mor(\mathbf{1}, (a \Rightarrow a) \Rightarrow a)$. Then for arbitrary objects a, the pair of objects $(a, \mathbf{1})$ and that of morphisms $(\mathcal{O}^a : a \to \mathbf{1}, Curry(Y)_a : \mathbf{1} \to (a \Rightarrow a) \Rightarrow a)$ defines a coercion pair. Take two objects a_1, a_2 and consider the corresponding coercion pairs. Then a morphism between them are determined by giving a morphism from a_1 to a_2. The morphism corresponding to the valid judgment $C,\ y : \beta \vdash y \cdot (\lambda x : \alpha.x) : \alpha$ and a pair $(a, \mathbf{1})$ with coercion morphisms as above is given by $Curry(1_a); Y_a : 1 \to a$. But, it was shown in [2] that the composition of Y and the polymorphic identity may not be dinatural with respect to $\mathcal{C}$. This shows that the family corresponding to the above judgment may not be dinatural with respect to $\mathcal{C}_C$.

4 Proof-Theoretic Analysis

In this section, we will consider dinaturality of $\lambda^{Top,\mathcal{L},\Rightarrow}_{\leq}$ in arbitrary ccc's. As we have seen, dinaturality may fail in general. But by using proof-theoretic idea, we can give a sufficinet condition for dinaturality of definable terms in arbitrary ccc's. This is an extension of a result by Girard-Scedrov-Scott [14].

We first need a few terminologies. A type in $\lambda^{Top,\mathcal{L},\Rightarrow}_{\leq}$ is called covariant (resp. contravariant) if all the occurrences of type variables are positive (resp. negative). A type is called uni-variant if it is covariant or contravariant and is called multi-variant if it is not uni-variant.

Definition 2 *The set $\mathcal{S}$ of $\lambda^{Top,\mathcal{L},\Rightarrow}_{\leq}$ types is defined inductively as follows:*

1. $Top \in \mathcal{S}$.

2. $\alpha \in \mathcal{S}$ *for any type variable* α.

3. $(s \Rightarrow t) \in \mathcal{S} \Leftrightarrow s$ *is uni-variant and* $t \in \mathcal{S}$.

4. $\{\ell_1 : t_1, \ldots, \ell_n : t_n\} \in \mathcal{S} \Leftrightarrow t_i \in \mathcal{S}$ *for all* $1 \leq i \leq n$.

For example, the type $(\alpha \Rightarrow \alpha) \Rightarrow \alpha$ does not belong to $\mathcal{S}$, for the type $\alpha \Rightarrow \alpha$ is multi-variant. The following theorem gives a sufficient condition on subtyping context for dinaturality in arbitrary ccc's.

Theorem 1 *Let* $C = \{\alpha_i \leq t_i | 1 \leq i \leq n\}$ *be a subtyping context and suppose that* $t_i \in \mathcal{S}$ *for* $1 \leq i \leq n$. *Then the image of the meaning function of a provable typing judgment* $C, A \vdash e : t$ *in* $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ *is a dinatural transformation between two functors* $\bar{s}^*, t^* : (\mathcal{C}_C)^\circ \times \mathcal{C}_C \to \mathcal{C}$, *where* $A = \{x_i : s_i | 1 \leq i \leq m\}$.

Thus, this theorem is applicable to the subtyping context $\{\alpha_1 \leq Top,\ \alpha_2 \leq \alpha_1,\ \alpha_3 \leq \alpha_1\}$ we discussed. But, it is not applicable to $\{\alpha \leq Top,\ \beta \leq (\alpha \Rightarrow \alpha) \Rightarrow \alpha\}$, which gave us a counter-example of dinaturality.

Our argument procceeds as follows. We first translate $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ into a simply-typed λ-calculus $\lambda^{\mathbf{1},\times,\Rightarrow}$ and then define a meaning function of the $\lambda^{\mathbf{1},\times,\Rightarrow}$ with the value in morphisms in ccc's. As the meaning function of the $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ factors as the composite of this translation from to $\lambda^{\mathbf{1},\times,\Rightarrow}$ and the meaning function on $\lambda^{\mathbf{1},\times,\Rightarrow}$, we can show dinaturality of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ by analyzing translated $\lambda^{\mathbf{1},\times,\Rightarrow}$ typing judgment by using cut-elimination procedure. To apply the cut-elimination procedure, we will adopt *the sequent-calculus formulation* [1, 14] of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ and $\lambda^{\mathbf{1},\times,\Rightarrow}$.

The language $\lambda^{\mathbf{1},\times,\Rightarrow}$ [19] is a simply-typed λ-calculus without subtyping. But we here allow the language to have proper axioms. Type expressions are defined by the following grammar:

$$t ::= \alpha | \mathbf{1} | t \Rightarrow t | t \times t$$

The type $\mathbf{1}$ corresponds to the terminal object of ccc's. The set of raw terms will be defined as follows:

$$e ::= c | * | x | \lambda x : t.e | e \cdot e | \langle e, e \rangle | \pi^1 e | \pi^2 e$$

The symbol $*$ denotes a constant of the type $\mathbf{1}$. The rules for the typing judgments and ones for the equations can be found in [1, 14].

We will translate $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ into $\lambda^{\mathbf{1},\times,\Rightarrow}$ by the method inspired by [4]. The trasnslation of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$ types to $\lambda^{\mathbf{1},\times,\Rightarrow}$ types is defined as follows.

$$\begin{aligned} \alpha_i^* &= \alpha_i \\ Top^* &= \mathbf{1} \\ (s \Rightarrow t)^* &= s^* \Rightarrow t^* \\ \{\ell_1 : t_1, \ldots, \ell_n : t_n\}^* &= t_1^* \times \ldots \times t_n^* \end{aligned}$$

A subtyping assertion $\alpha \leq t$ in C will be translated into a proper axiom $x : \alpha \vdash f_\alpha(x) : t^*$, where f_α is a constant. We denote by C^* the set of proper axioms attached to the subtyping assertions in C. A derivation σ of subtyping judgments $C \vdash s \leq t$ will be translated into a $\lambda^{\mathbf{1},\times,\Rightarrow}$judgment of the form $x : s^* \vdash P : t^*$ [4]. Then we can show that this typing judgment $x : s^* \vdash P : t^*$ is a valid derivation from C^* in $\lambda^{\mathbf{1},\times,\Rightarrow}$and that all translations of derivations of a judgment in $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$are provably equivalent in $\lambda^{\mathbf{1},\times,\Rightarrow}$ [4].

We will translate a $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$typing judgment of the form $C, A \vdash e : t$ into a $\lambda^{\mathbf{1},\times,\Rightarrow}$typing judgment of the form $A^* \vdash e^* : t^*$, where $A^* = \{x_i : s_i^* | 1 \leq i \leq m\}$ for $A = \{x_i : s_i | 1 \leq i \leq m\}$. Also in this case, we first translate derivations of typing judgments in $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$to those in $\lambda^{\mathbf{1},\times,\Rightarrow}$. As the main difference between $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$and $\lambda^{\mathbf{1},\times,\Rightarrow}$lies in the rules for subtyping judgments and subsumption rule, we only have to discuss the subsumption rule. An instance of (*subsumption*)

$$\frac{C, A \vdash e : s \quad C \vdash s \leq t}{C, A \vdash e : t}$$

is replaced by a cut

$$\frac{A^* \vdash e^* : s^* \quad x : s^* \vdash P : t^*}{A^* \vdash [e^*/x]P : t^*}$$

where $A^* \vdash e^* : s^*$ and $x : s^* \vdash P : t^*$ are the typing judgments corresponding to $C, A \vdash e : s$ and $C \vdash s \leq t$ respectively. If a typing judgment $C, A \vdash e : s$ is derivable in $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$, the corresponding typing judgment $A^* \vdash e^* : s^*$ is a valid derivation in $\lambda^{\mathbf{1},\times,\Rightarrow}$from C^* and all translations of derivations of a judgment in $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$are provably equivalent in $\lambda^{\mathbf{1},\times,\Rightarrow}$ [4].

We can attach morphisms in a ccc to typing judgments in $\lambda^{\mathbf{1},\times,\Rightarrow}$ [1, 14]. Here, given a coercion pair $(\mathbf{a}, \mathbf{f})$, we attach to a proper axiom $x : \alpha_i \vdash f_{\alpha_i}(x) : t_i^*$ the morphism $f_i : a_i \to t_i^*\mathbf{aa}$. Hence we can attach to a derivation of $A^* \vdash e : t^*$ a family of morphisms $[\![e]\!]_{\mathbf{a},\mathbf{f}} : \bar{s}^*\mathbf{aa} \to t^*\mathbf{aa}$ for $(\mathbf{a}, \mathbf{f}) \in Obj(\mathcal{C}_C)$. It is apparent from the construction that the meaning function of $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$factors as the composite of the translation into $\lambda^{\mathbf{1},\times,\Rightarrow}$and the meaning function of $\lambda^{\mathbf{1},\times,\Rightarrow}$. Thus, to prove dinaturality of the $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$term, we only have to consider the corresponding $\lambda^{\mathbf{1},\times,\Rightarrow}$term.

Now we apply the standard proof-theoretic armament, cut-elimination, to translated $\lambda^{\mathbf{1},\times,\Rightarrow}$typing judgment so that we obtain dinaturality of the calculus $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$. It is easy to see that the cut-elimination procedure leaves the meaning of the proof unchanged. As we have introduced proper axioms, we cannot expect all cuts be eliminated. But, by applying the procedure, we can obtain a derivation that is convenient for the establishment of parametricity under the condition on subtyping context stated in the above theorem. A similar technique has been employed in [12, 13].

But, before applying the cut-elimination procedure, we simplify the set of proper axioms. The simplification goes as follows: If a proper axiom is of the form $x : \alpha \vdash f_\alpha(x) : s \times t$, then we replace it with two axioms $\{x : \alpha \vdash \pi^1 f_\alpha(x) : s, \quad x : \alpha \vdash \pi^2 f_\alpha(x) : t\}$. If a proper axiom is of the form $x : \alpha \vdash f_\alpha(x) : s \Rightarrow t$, we replace it with $x : \alpha, y : s \vdash f_\alpha(x) \cdot y : t$. If the subtyping context C satisfies the condition mentioned in the above theorem, then, by repeatedly applying this procedure, we eventually obtain a set C^{**} of

proper axioms of the form $y_1 : s_1, \ldots, y_m : s_m \vdash g(y_1, \ldots y_m) : t$, where s_1 is a type variable, t is a type variable or $\mathbf{1}$, and all $s_2, \ldots s_m$ are uni-variant. And this set C^{**} and the original set C^* are (essentially) equiprovable. We further can show that the morphisms corresponding to these axioms in C^{**} are dinatural with respect to $\mathcal{C}_C$. Thus, we can consider the translated $\lambda^{\mathbf{1},\times,\Rightarrow}$judgment as derived from the set of proper axioms in which only uni-variant types occur. By applying cut-elimination procedure to $\lambda^{\mathbf{1},\times,\Rightarrow}$judgments, we can show that all multi-variant cuts are eliminable. And we can easily see that, from the definition, cuts at uni-variant types preserve dinaturality. This concludes our theorem.

Let us consider how dinaturality may fail if the subtyping context does not satisfy the above condition. For the concreteness, let us consider our familiar case that $C = \{\alpha \leq Top,\ \ \beta \leq (\alpha \Rightarrow \alpha) \Rightarrow \alpha\}$. Then the corresponding set of proper axioms is given as $C^* = \{x : \alpha \vdash f_\alpha(x) : \mathbf{1},\ \ y : \beta \vdash f_\beta(y) : (\alpha \Rightarrow \alpha) \Rightarrow \alpha\}$. In this case, we have $C^{**} = \{x : \alpha \vdash f_\alpha(x) : \mathbf{1},\ \ y : \beta, z : \alpha \Rightarrow \alpha \vdash f_\beta(y) \cdot z : \alpha\}$. Let us consider the $\lambda_{\leq}^{Top,\mathcal{L},\Rightarrow}$typing judgment $C, y : \beta \vdash y \cdot (\lambda x : \alpha.x) : \alpha$. Then the corresponding derivation in $\lambda^{\mathbf{1},\times,\Rightarrow}$is

$$\frac{\vdash \lambda x : \alpha.x : \alpha \Rightarrow \alpha \quad y : \beta, z : \alpha \Rightarrow \alpha \vdash f_\beta(y) \cdot z : \alpha}{y : \beta \vdash f_\beta(y) \cdot (\lambda x : \alpha.x) : \alpha}$$

Here the cut occurs at a multi-variant type $\alpha \Rightarrow \alpha$ and is not eliminable.

5 Conclusion and Future Work

In this article, we exhibited a formulation of semantic parametricity of the calculus with simple subtyping and its accompanying problems. Based on the view that "subtyping as implicit coercion," our formulation of dinatural transformations nicely extends that of the calculus without subtyping. And we have given a sufficient condition for dinaturality, which seems useful for actual use of subtyping. But there are also many technical challenges left unanswered.

We have seen that dinaturality may fail in arbitrary ccc's. The obvious question is whether there exists a ccc such that the definable terms have dinaturality with respect to all subtyping contexts. In particular, we are very interested in whether this holds in **PER** or not, for **PER** has been the most successful model of polymorphism and parametricity. (A recent paper of Freyd-Robinson-Rosolini [11] suggests significant inconvenience of the category for the study of parametricity, though.) On the other hand, in this article we allowed arbitrary coercion morphisms. The coercion pair used in the explanation of the failure of dinaturality does not look like a morphism naturally associated with subtyping. Bruce-Longo [5] and Breazu-Tannen-Coquand-Gunter-Scedrov [4] suggests appropriateness of considering coercion morphisms of specific type. For example, in Bruce-Longo [5], only realizable morphisms whose Gödel numbers are identical to that of identity function were allowed as coercion morphisms. But it is unclear whether this restriction is helpful to establish dinaturality of our calculus in **PER**.

We have only disscussed a simply-typed λ-calculus. As explicit bounded quantification introduces further challenges, it seems immature for us to investigate it right now. But dinatural calculus may also

bring opportunities, especially beyond pure λ-calculus. For example, let us consider a update-function [5] *inc* which increments the value of the label ℓ. Let $A = \{\ell : int, \ldots\}, B = \{\ell : int, \ldots\}$ and f be an arbitrary function from A to B that leaves the value corresponding to the label ℓ unchanged. Then we expect that *inc* satisfies an equality $f; inc = inc; f$, which, intuitively speaking, can be viewed as a kind of "parametricity." Although such a function is not definable in pure λ-calculus [5], it may be possible to define it as a dinatural transformation with respect to some category of coercions. And our category-theoretic setting may also be applicable to the study of F-bounded polymorphism [6], which reflects another crucial feature of object-oriented languages. Of course, in order to discuss these issues characteristic of typed object-oriented languages, we need further understanding of dinaturality of pure λ-calculus.

References

[1] A. Asperti and G. Longo, *Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist*, MIT Press, 1991.

[2] E. S. Bainbridge, P. Freyd, A. Scedrov and P. J. Scott, Functorial polymorphism, *Theoretical Computer Science*, 70, 35–64, 1990.

[3] R. Blute, Linear logic, coherence and dinaturality, to appear in *Theoretical Computer Science.*

[4] V. Breazu-Tannen, T. Coquand, C. A. Gunter and A. Scedrov, Inheritance as implicit coercion, *Information and Computation*, 93, 172–221, 1991.

[5] K. Bruce and G. Longo, A modest model of records, inheritance, and bounded quantification, *Information and Computation*, 87, 1/2, 196–240, 1990.

[6] P. Canning, W. Cook, W. Hill, J. C. Mitchell and W. Olthoff, F-bounded quantification for object-oriented programming, in *Proceedings of Fourth International Confrence on Functional Programming Languages and Computer Architecture*, 273–280, ACM, 1989.

[7] L. Cardelli, S. Martini, J. C. Mitchell and A. Scedrov, An extension of system F with subtyping, in T. Ito and A. R. Meyer (eds) *TACS '91*, Lecture Notes in Computer Science 526, 750–770, Springer Verlag, 1991.

[8] L. Cardelli and J. C. Mitchell, Operations on records, *Mathematical Structures in Computer Science*, 1, 1, 3–48, 1991.

[9] L. Cardelli and P. Wegner, On understanding types, data abstraction, and polymorphism, *ACM Comuting Surveys*, 17, 471–522, 1985.

[10] P. -L. Curien and G. Ghelli, Coherence of subsumption, in A. Arnold (ed) *CAAP'90*, Lecture Notes in Computer Science 431, 132–146, Springer Verlag, 1990.

[11] P. J. Freyd, E. P. Robinson and G. Rosolini, Functorial parametricity, in *Proc. of 7-th IEEE Symposium on Logic in Computer Science*, 444–452, IEEE, 1992.

[12] V. Gehlot and C. Gunter, Normal process representatives, in *Proc. of 5-th IEEE Symposium on Logic in Computer Science*, 200–207, IEEE, 1990.

[13] J. -Y. Girard, Y. Lafont and P. Taylor, *Proofs and Types*, Cambridge University Press, 1989.

[14] J. -Y. Girard, A. Scedrov, and P. J. Scott, Normal forms and cut-free proofs as natural transformations, in Y. N. Moschovakis (ed) *Logic from Computer Science*, Springer Verlag, 1992.

[15] J. Lambek and P. J. Scott, *Introduction to Higher Order Categorical Logic*, Cambridge University Press, 1986.

[16] Q. Ma, Parametricity as subtyping (preliminary report), in *Proc. of 19-th ACM Symp. on Principles of Programming Languages*, 281–292, 1992.

[17] S. MacLane, *Categories for the Working Mathematician*, Springer Verlag, 1971.

[18] J. C. Mitchell, Toward a typed foundation for method specialization and inheritance, in *Proc. of 17-th ACM Symp. on Principles of Programming Languages*, 109–124, 1990.

[19] J. C. Mitchell, Type systems for programming languages, in J. van Leeuwen (ed) *Handbook of Theoretical Computer Science Vol. B*, 365–458, 1990, The MIT Press/Elsevier.

[20] J. C. Mitchell, Type inference with simple subtypes, *Journal of Functional Programming*, 1, 3, 245–285, 1991.

[21] J. C. Mitchell and A. Meyer, Second-order logical relations (extended abstract), in R. Parikh (ed) *Logic of Programs*, Lecture Notes in Computer Science 193, 225–236, Springer Verlag, 1985.

[22] G. D. Plotkin and M. Abadi, A logic for parametric polymorphism, in M. Bezem and J. F. Groote (eds) *Proc. of the International Conference on Typed Lambda Calculi and Applications*, Lecture Notes in Computer Science 664, Springer Verlag, 1993.

[23] J. C. Reynolds, Using category theory to design implicit conversions and generic operators, in N. D. Jones (ed) *Semantics-Directed Compiler Generation*, Lecture Notes in Computer Science 94, 211–258, Springer Verlag, 1980.

[24] J. C. Reynolds, Types, abstraction and parametric polymorphism, in R. E. A. Mason (ed), *Information Processing 83*, 513–523, North-Holland, 1983.

[25] J. C. Reynolds, Polymorphism is not set-theoretic, in G. Kahn, D. B. MacQueen and G. Plotkin (eds), *Semantics of Data Types*, Lecture Notes in Computer Science 173, 145–156, Springer Verlag, 1984.

[26] P. Wadler, Theorems for free!, in *Proceedings of Fourth International Confrence on Functional Programming Languages and Computer Architecture*, 347–359, ACM, 1989.