

1995-02

Equational Axiomatization of Bicoercibility for Polymorphic Types

Tiurny, Jerzy. "Equational Axiomatization of Bicoercibility for Polymorphic Types", Technical Report BUCS-1995-004, Computer Science Department, Boston University, February 16, 1995. [Available from: <http://hdl.handle.net/2144/1565>]

<https://hdl.handle.net/2144/1565>

"Downloaded from OpenBU. Boston University's institutional repository."

Equational Axiomatization of Bicoercibility for Polymorphic Types

Jerzy Tiuryn*
Institute of Informatics
Warsaw University
Banacha 2
02-097 Warsaw, POLAND
(tiuryn@mimuw.edu.pl)

February, 1995

Abstract

Two polymorphic types σ and τ are said to be bicoercible if there is a coercion from σ to τ and conversely. We give a complete equational axiomatization of bicoercible types and prove that the relation of bicoercibility is decidable.

1 Introduction

The notion of a *subtype* of a type plays an important role in typed programming languages and it has been a subject of an intensive research recently [Ben93, BCGS91, CMSS94, CGL92, CP94, LMS94, Mit90, Tiu92, TW93]. There is a number of fundamental open problems which have to be solved in order to gain a better understanding of the notion of a subtype. One of such problems is the question of decidability of the relation of subtyping

*This work is partly supported by NSF Grant CCR-9113196, KBN Grant 2 P301 031 06 and by ESPRIT BRA7232 GENTZEN.

for second-order polymorphic types. This relation has been axiomatized by John Mitchell [Mit90]. The fact that a type σ is a subtype of a type τ can be established by finding a *coercion* from σ to τ . Coercions are denoted by terms, typable in system F of polymorphic second-order lambda calculus. The important feature of coercions is that after erasing all the type information they are $\beta\eta$ equal to the identity. Despite this simple computational behaviour it is not known how to decide for two given polymorphic types σ and τ whether there is a coercion from σ to τ . Part of the difficulty in proving this problem decidable is the axiom which says that the type $\forall X. \sigma$ is a subtype of each of its instances. In symbols:

$$\forall X. \sigma \leq [\tau/X]\sigma$$

holds for every type τ . The above subtype relation is established by the coercion $\lambda x : \forall X. \sigma \ x[\tau] : \forall X. \sigma \rightarrow [\tau/X]\sigma$.

In the present paper we are concerned with the notion of a bicoercion. Call σ and τ *bicoercible* iff there are coercions from σ to τ and conversely from τ to σ . The main result of the paper is a complete equational axiomatization of the relation of bicoercibility. It turns out that this equational theory is decidable, implying therefore decidability of the problem of bicoercibility for polymorphic types.

The relation of bicoercibility for polymorphic types is a special kind of isomorphism. This observation follows from the *coherence property* (see [LMS94]), *i.e.* the property which says that if M and N are coercions from σ to τ , then they are provably equal in a certain extension of the system F . Bicoercibility of polymorphic types neither contains nor is contained in the notion of a provable isomorphism in system F . The reader is referred to [Dic95] for characterization of the latter notion. For example, $\forall Y. \forall X. X$ and $\forall X. X$ are bicoercible but they are not provably isomorphic in F . On the other hand, $X \rightarrow (Y \rightarrow Z)$ and $Y \rightarrow (X \rightarrow Z)$ are provably isomorphic in F but they are not bicoercible.

The paper is organized as follows. In Section 2 we recall the Mitchell's system of subtyping for polymorphic types. We also define there a rewrite system which is very much related to the Mitchell's proof system. Rewriting σ into τ results in establishing that σ is a subtype of τ . In Section 3 we introduce the equational system for deriving bicoercions. This system is

decidable. Sections 4 and 5 are devoted to the proof of its completeness. Section 4 collects some technical results which show some kind of a control which we have on moving quantifiers in types when rewriting them. This part of the proof deals with a more general situation of arbitrary coercions, rather than bicoercions. The proof of completeness is concluded in Section 5 where we characterize the so called *reversible rewrite steps* — steps which are used when deriving a bicoercion.

2 Subtyping for Polymorphic Types

First we present the Mitchell's system of subtyping for polymorphic types (see [Mit90]). The system derives formulas of the form $\sigma \leq \tau$, where σ and τ are polymorphic types.

Axioms:

- (refl) $\sigma \leq \sigma$
- (inst) $\forall X. \sigma \leq [\rho/X]\sigma$
- (dummy) $\sigma \leq \forall X. \sigma$, X doesn't occur free in σ .
- (distr) $\forall X. (\sigma \rightarrow \tau) \leq \forall X. \sigma \rightarrow \forall X. \tau$

Rules:

$$(\rightarrow) \frac{\sigma' \leq \sigma \quad \tau \leq \tau'}{\sigma \rightarrow \tau \leq \sigma' \rightarrow \tau'} \qquad (\forall) \frac{\sigma \leq \tau}{\forall X. \sigma \leq \forall X. \tau}$$

$$(\text{trans}) \frac{\sigma \leq \rho \quad \rho \leq \tau}{\sigma \leq \tau}$$

We write $\vdash \sigma \leq \tau$ to indicate that $\sigma \leq \tau$ is derivable in the above system.

In the rest of the paper we will view polymorphic types as binary trees such that:

- inner nodes are labelled \rightarrow ;
- leaves are labelled with type variables X, Y, \dots ;
- every node is labelled by a finite sequence of quantified variables $\forall X_1 \forall X_2 \dots$

A node in a type σ will be identified with a path (a sequence of 0's and 1's) leading from the root to that node. A node w is said *positive* if the number of 0's in w is even. Otherwise w is called *negative*.

We define a rewrite system for polymorphic types, closely related to the above formal system. We say that σ *rewrites* into τ in one step, denoted $\sigma \sqsubset \tau$, if τ is obtained from σ by applying one of the following rewrite rules:

For a positive node perform one of the following steps:

- (*p-inst*) $\quad \forall X. \sigma \sqsubset [\rho/X]\sigma$
- (*p-dummy*) $\quad \sigma \sqsubset \forall X. \sigma, \quad X \text{ doesn't occur free in } \sigma.$
- (*p-distr*) $\quad \forall X. (\sigma \rightarrow \tau) \sqsubset \forall X. \sigma \rightarrow \forall X. \tau$

For a negative node perform one of the following steps:

- (*n-inst*) $\quad [\rho/X]\sigma \sqsubset \forall X. \sigma$
- (*n-dummy*) $\quad \forall X. \sigma \sqsubset \sigma, \quad X \text{ doesn't occur free in } \sigma.$
- (*n-distr*) $\quad \forall X. \sigma \rightarrow \forall Y. \tau \sqsubset \forall Z. ([Z/X]\sigma \rightarrow [Z/Y]\tau), \quad Z \text{ is a new variable.}$

When performing steps (*p-inst*) and (*n-inst*) a care has to be taken in order to avoid quantifier clashes — rename bound variables, when necessary to prevent such clashes.

3 Bicoercion

Two types σ and τ are said to be *bicoercible* if $\vdash \sigma \leq \tau$ and $\vdash \tau \leq \sigma$ holds. We start with some examples of bicoercible types.

Lemma 1 *Types $\forall X \forall Y. \sigma$ and $\forall Y \forall X. \sigma$ are bicoercible.*

Proof: We prove the \leq inequality.

$$\frac{\frac{\frac{\forall Y. \sigma \leq \sigma}{\forall X \forall Y. \sigma \leq \forall X. \sigma} (\forall)}{\forall Y \forall X \forall Y. \sigma \leq \forall Y \forall X \forall Y. \sigma} (\forall)}{\forall X \forall Y. \sigma \leq \forall Y \forall X. \sigma} (\text{trans})$$

The opposite inequality follows by symmetry of assumptions. ■

Lemma 2 *If every occurrence of X in σ is positive, then $\forall X. \sigma$ and $[(\forall X. X)/X]\sigma$ are bicoercible.*

Proof: The \leq inequality is just the (inst) axiom. For the proof of \geq we first show the following fact which is very easy to establish by mutual induction on σ and τ (we omit the proof). If X occurs only positively in σ and only negatively in τ , then

$$[(\forall X. X)/X]\sigma \leq \sigma \quad \text{and} \quad \tau \leq [(\forall X. X)/X]\tau$$

Now, having the above we proceed as follows.

$$\frac{\frac{[(\forall X. X)/X]\sigma \leq \sigma}{[(\forall X. X)/X]\sigma \leq \forall X. [(\forall X. X)/X]\sigma} \quad \frac{[(\forall X. X)/X]\sigma \leq \sigma}{\forall X. [(\forall X. X)/X]\sigma \leq \forall X. \sigma} (\forall)}{\frac{[(\forall X. X)/X]\sigma \leq \forall X. [(\forall X. X)/X]\sigma}{[(\forall X. X)/X]\sigma \leq \forall X. \sigma} (\text{trans})}$$

■

Lemma 3 *If X doesn't occur free in σ , then $\forall X. (\sigma \rightarrow \tau)$ and $\sigma \rightarrow \forall X. \tau$ are bicoercible.*

Proof: For \leq we have the following derivation.

$$\frac{\forall X. (\sigma \rightarrow \tau) \leq \forall X. \sigma \rightarrow \forall X. \tau \quad \frac{\sigma \leq \forall X. \sigma \quad \forall X. \tau \leq \forall X. \tau}{\forall X. \sigma \rightarrow \forall X. \tau \leq \sigma \rightarrow \forall X. \tau} (\rightarrow)}{\forall X. (\sigma \rightarrow \tau) \leq \sigma \rightarrow \forall X. \tau} (\text{trans})$$

For the proof of \geq we have the following derivation.

$$\frac{\sigma \rightarrow \forall X. \tau \leq \forall X. (\sigma \rightarrow \forall X. \tau) \quad \frac{\frac{\forall X. \tau \leq \tau \quad \sigma \leq \sigma}{\sigma \rightarrow \forall X. \tau \leq \sigma \rightarrow \tau} (\rightarrow)}{\forall X. (\sigma \rightarrow \forall X. \tau) \leq \forall X. (\sigma \rightarrow \tau)} (\forall)}{\sigma \rightarrow \forall X. \tau \leq \forall X. (\sigma \rightarrow \tau)} (\text{trans})$$

■

3.1 Proof System

The system given below is for deriving expressions of the form $\sigma \equiv \tau$, where σ and τ are polymorphic types.

Axioms:

(A1) $\sigma \equiv \sigma$

(A2) $\forall X \forall Y. \sigma \equiv \forall Y \forall X. \sigma$

(A3) $\forall X. \sigma \equiv [(\forall X. X)/X]\sigma$, all occurrences of X in σ are positive.

(A4) $\forall X. (\sigma \rightarrow \tau) \equiv \sigma \rightarrow \forall X. \tau$, X doesn't occur free in σ , and it has a negative occurrence in τ .¹

Rules:

$$\begin{array}{cc}
 \text{(arrow)} \quad \frac{\sigma \equiv \sigma' \quad \tau \equiv \tau'}{\sigma \rightarrow \sigma' \equiv \tau \rightarrow \tau'} & \text{(quant)} \quad \frac{\sigma \equiv \sigma'}{\forall X. \sigma \equiv \forall X. \sigma'} \\
 \\
 \text{(trans)} \quad \frac{\sigma \equiv \rho \quad \rho \equiv \tau}{\sigma \equiv \tau} & \text{(symm)} \quad \frac{\sigma \equiv \tau}{\tau \equiv \sigma}
 \end{array}$$

We write $\vdash \sigma \equiv \tau$ to indicate that there is a derivation of $\sigma \equiv \tau$ in the above proof system.

The main result of this paper is the following theorem. It shows that the above proof system captures bicoercibility.

Theorem 4 σ and τ are bicoercible iff $\vdash \sigma \equiv \tau$ holds.

Let us observe that the implication (\Leftarrow) is *soundness* of our proof system, while the implication (\Rightarrow) expresses its *completeness*. Proof of the above theorem will be given at the end of Section 5. We conclude this section with a corollary of the above result, and with a technical lemma which will be used later.

Corollary 5 *It is decidable for two given polymorphic types whether they are bicoercible.*

Proof: Given two types σ and τ . Use axioms (A3) and (A4) as rewrite rules (ordered from left to right) and move the quantifiers in these types as far as possible, using (A2) to permute quantifiers when necessary. If the obtained two types differ only with respect to the order of quantifiers and with respect to α -conversion (names of bound variables)², then $\vdash \sigma \equiv \tau$

¹If X occurs in τ only positively, then this case is handled by (A3). We have added this constraint to (A4) in order to simplify the proof of Lemma 6.

²Is is easy to decide these two properties of polymorphic types.

holds. Otherwise $\not\vdash \sigma \equiv \tau$. Then Theorem 4 yields correctness of the above algorithm as a test for bicoercibility. ■

Lemma 6 *If $\vdash \sigma \rightarrow \sigma' \equiv \tau \rightarrow \tau'$ holds, then both $\vdash \sigma \equiv \tau$ and $\vdash \sigma' \equiv \tau'$ hold.*

Proof: We prove the lemma by induction on the number of steps in derivation of $\sigma \rightarrow \sigma' \equiv \tau \rightarrow \tau'$. The only slightly non-trivial step is when the last rule used in the derivation is (trans). Then, for a certain type ρ we have

$$\vdash \sigma \rightarrow \sigma' \equiv \rho \quad \text{and} \quad \vdash \rho \equiv \tau \rightarrow \tau'$$

If ρ is of the form $\rho' \rightarrow \rho''$, then we are done (apply the induction assumption). Otherwise ρ starts with a quantifier.³ Hence $\rho \equiv \tau \rightarrow \tau'$ and $\rho \equiv \sigma \rightarrow \sigma'$ are instances of an axiom (A3) or (A4). Since these axioms are applicable in disjoint situations, it follows that both $\rho \equiv \tau \rightarrow \tau'$ and $\rho \equiv \sigma \rightarrow \sigma'$ must be instances of the same axiom. Hence $\sigma \rightarrow \sigma' = \tau \rightarrow \tau'$, i.e. $\sigma = \tau$ and $\sigma' = \tau'$, and we are done. ■

4 Auxiliary Results

We first introduce the concept of a *marking*. Let σ be a type and let X be a variable which occurs in σ . This occurrence is marked 0 if it is free, it is marked -1 if it is bound by a quantifier which is at a negative node, and marked $+1$ if it is bound by a quantifier which is at a positive node.

The following result expresses the property that -1 marks are “easy to create” but impossible to “get rid off”.

Lemma 7 *Let $\sigma \sqsubset^* \tau$ and let w be a leaf in σ . Assume that one of the following three conditions holds.*

(i) *w doesn't belong to τ .*

³Obviously it cannot be a type variable.

(ii) w is marked in σ with -1 .

(iii) w is marked in σ with 0 , and it is a leaf in τ marked $+1$ or -1 .

Then there is $u \leq w$ such that u is a leaf in τ marked -1 .

Proof: We prove the lemma by induction on the number of steps in deriving $\sigma \sqsubset^* \tau$. Let's consider first a one step $\sigma \sqsubset^* \tau$. If w doesn't belong to τ , then this must have resulted from an application of (*n-inst*) rule at node $w' \leq w$. Then w' is a negative node and this step introduces a quantifier at this node, which binds a variable at a leaf $u \leq w$. Thus u is marked -1 . Assume now that w belongs to τ . If w is marked in σ with 0 or -1 , then w must be a leaf in τ (since no step substituting a type for w in σ can be performed in this case). It follows that if w is marked -1 in σ , then the same mark stays at w in τ . Now, if w is marked 0 in σ but w is bound in τ , then it means that this step must have introduced a quantifier at an existing node. This is only possible at (*n-inst*) step and this means that w is marked -1 in τ . This completes the base step of induction.

Let's assume now that $\sigma \sqsubset \rho \sqsubset^* \tau$. If w doesn't belong to ρ , then by the previous analysis, it follows that there is $w' \leq w$ such that w' is a leaf in ρ marked -1 . Thus, by induction assumption applied to $\rho \sqsubset^* \tau$ and the node w' we get the conclusion.

Assume now that w belongs to ρ . If w doesn't belong to τ , then take any w' which extends w and is a leaf in ρ . Clearly w' doesn't belong to τ either. By induction assumption applied to $\rho \sqsubset^* \tau$ and the node w' we get the conclusion (since w doesn't belong to τ , it follows that the leaf u in τ such that $u \leq w'$ must satisfy $u \leq w$).

To complete the proof let's assume that w is marked in σ with 0 or -1 and w belongs to ρ . It follows from the analysis of the base case that then w is a leaf in ρ marked 0 or -1 . Thus, by induction assumption applied to $\rho \sqsubset^* \tau$ and the node w we get the conclusion. This completes the proof of lemma..

■

The next result shows that moving a quantifier downwards from a positive node to a positive node can be done only in a very limited way.

Lemma 8 *Let $\sigma \sqsubset^* \tau$ and let X be a variable which occurs in both: σ and τ at the same node, say u . Let us assume that this occurrence of X in σ is bound by a quantifier at a positive node w_σ and in τ at a positive node w_τ . If $w_\sigma \leq w_\tau$, then $w_\tau = w_\sigma 1^m$, for some $m \geq 0$.*

Proof: We prove the lemma by induction on the number of steps in getting from σ to τ . If $\sigma \sqsubset \tau$ and the quantifier moves down from a positive node to a positive node then it is only possible by performing (*p-distr*) step. The clearly we obtain $w_\tau = w_\sigma 1$, as required.

Now, let us assume that $\sigma \sqsubset \rho \sqsubset^* \tau$, and let u be the node at which the variable X occurs. If u is not a leaf in ρ , then by Lemma 7 (i) and (ii) the node u in τ must have been marked -1 , contradiction. Applying Lemma 7 again we conclude that u must be marked in ρ with $+1$. Let w_ρ be the node in ρ at which the binding quantifier for u is located. It follows from the analysis of the base case that either $w_\rho = w_\sigma$, or $w_\rho = w_\sigma 1$. If $w_\sigma \leq w_\tau$ and former possibility holds, then, by induction assumption, we get that $w_\tau = w_\rho 1^m$, for some $m \geq 0$. Hence $w_\tau = w_\sigma 1^m$. If $w_\sigma \leq w_\tau$ and latter possibility holds, then either $w_\tau = w_\sigma$ and we are done, or else $w_\rho \leq w_\sigma$ (since $w_\rho \leq u$ and $w_\tau \leq u$), and again by induction assumption we conclude that $w_\tau = w_\rho 1^m$, for some $m \geq 0$, i.e. $w_\tau = w_\sigma 1^{m+1}$, for some $m \geq 0$. This completes the proof. ■

Let u_1 and u_2 be leaves in a type σ . A positive occurrence of a quantifier $\forall X$ in σ which binds a variable at nodes u_1 and u_2 is said to be *splitable* at nodes u_1 and u_2 , if there exists a type τ and two positive nodes w_1 and w_2 in τ such that $\sigma \sqsubset^* \tau$ and u_1 and u_2 are leaves in τ bound by two different quantifiers: one at node w_1 and the other at node w_2 . The case $w_1 = w_2$ is allowed. We call every pair (σ, τ) with the above properties *splitting pair*.

Lemma 9 *If $\forall X$ is splitable in σ at nodes u_1 and u_2 , then u_1 and u_2 are positive.*

Proof: We prove the lemma by induction on the number of steps in deriving $\sigma \sqsubset^* \tau$ such that (σ, τ) is a splitting pair. By inspecting all the six cases it is easy to verify that in one step one cannot get a splitting pair.

Let us suppose that $\sigma \sqsubset \rho \sqsubset^* \tau$ and (σ, τ) is a splitting pair at nodes u_1 and u_2 . If (ρ, τ) is also a splitting pair at nodes u_1 and u_2 (perhaps for a quantifier located in a position different than $\forall X$ in σ) then, by induction assumption, we conclude that u_1 and u_2 are positive.

Consider now the situation when (ρ, τ) is not a splitting pair for u_1 and u_2 . If u_j is not contained in ρ (where $j = 1$ or 2), then by Lemma 7 (i), there is $u'_j \leq u_j$ such that u'_j is a leaf in ρ , marked with -1 . Since $\rho \sqsubset^* \tau$, it follows Lemma 7 (ii) that there is $u''_j \leq u'_j$ such that u''_j is a leaf in τ marked -1 . This is a contradiction since u_j is a leaf in τ marked $+1$. Thus both: u_1 and u_2 belong to ρ .

If u_j is not a leaf, then by Lemma 7 (ii) we would get a contradiction since the marking of u_j in τ is $+1$. Thus u_1 and u_2 are leaves in ρ .

If the variable at u_j is marked -1 or 0 , then again, by Lemma 7 (ii) or (iii) we get a contradiction with the marking of u_j in τ . Thus both u_1 and u_2 are variables in ρ , marked $+1$, and they are bound by two different quantifiers.

Let's analyse now the first step $\sigma \sqsubset \rho$ to see when we can ρ with the above properties.. Clearly it cannot be (*p-dummy*) or (*n-dummy*) . Let w be the node in σ at which the $\forall X$ occurs. Consider the following remaining four cases.

Case I: (*p-distr*)

The only possibility in this case is that (*p-distr*) is performed at w for $\forall X$ and $u_1 \leq w0$ and $u_2 \leq w1$. Then u_1 would be marked -1 in ρ , a contradiction. Thus this case is impossible.

Case II: (*n-distr*)

This case is clearly impossible.

Case III: (*p-inst*)

Then the quantifier $\forall X$ at node w must have been instantiated. Obviously it must have been instantiated with the type of the form $\forall Y_1 \dots \forall Y_n. Y_i$, where $1 \leq i \leq n$. Thus, in order to conform to the requirement that both u_1 and u_2 are marked $+1$ in ρ — it follows that both u_1 and u_2 must be positive.

Case IV: (*n-inst*)

If the quantifier introduced by this step binds u_i (for $i = 1$ or 2), then u_i

would have been marked in ρ with -1 . This is impossible.

This completes the proof of the lemma. ■

5 Reversible Steps

Call one step $\sigma \sqsubset \tau$ *reversible* if $\tau \sqsubset^* \sigma$ holds. In the next sequence of lemmas we will characterize all reversible steps.

Lemma 10 (p-dummy)

(p-dummy) step is always reversible.

Proof: Obvious. ■

Lemma 11 (p-distr)

Let w be a positive path in σ and let $\sigma \sqsubset \tau$ by a (p-distr) step performed at node w by moving the quantifier $\forall X$ to w_0 and w_1 . This step is reversible iff X doesn't occur free at node w_0 .

Proof: If X occurs free below w_0 , then this occurrence is marked -1 in τ , while it is marked $+1$ in σ . Thus, by Lemma 7 (ii), the relation $\tau \sqsubset^* \sigma$ is impossible. Obtained contradiction proves the result. ■

Lemma 12 (p-inst)

Let w be a positive path in σ and let $\sigma \sqsubset \tau$ by a (p-inst) step performed at node w by instantiating a quantifier $\forall X$ with a type ρ . This step is reversible iff one of the following conditions holds:

- (i) $\forall X$ is dummy.
- (ii) $\forall X$ is not dummy, all bindings of $\forall X$ are positive and ρ is of the form $\forall Y_1 \dots \forall Y_n. Y_i$, for some $1 \leq i \leq n$.

(iii) $\forall X$ is not dummy, there is another quantifier $\forall Z$ at a node $u \leq w$, such that $w = u1^m$, for some $m \geq 0$; $\forall Z$ has only positive bindings; and $\forall X$ is in scope of $\forall Z$. Type ρ is of the form $\forall Y_1 \dots \forall Y_n. Z$, where $Z \notin \{Y_1, \dots, Y_n\}$. Moreover, if $\forall X$ has a negative binding, then $\forall Z$ is dummy.

Proof: Let us first observe that each of the above conditions implies reversibility of the (*p-inst*) step. In case of (ii) as well as in case of (iii), when both $\forall X$ and $\forall Z$ have only positive bindings, then it follows from Lemma 2. In case of (iii) when $\forall Z$ is dummy one gets back from τ to σ by a sequence of (*p-distr*) steps.

We prove now that if the (*p-inst*) step is reversible then one of the (i)–(iii) must hold. Assume that $\forall X$ is not dummy. If depth of ρ is greater than 1, then, by Lemma 7 we cannot get $\tau \sqsubset^* \sigma$. Hence ρ is of one of the following two forms:

$$\rho = \forall Y_1 \dots \forall Y_n. Y_i \tag{1}$$

or

$$\rho = \forall Y_1 \dots \forall Y_n. Z \tag{2}$$

where $1 \leq i \leq n$ and $Z \notin \{Y_1, \dots, Y_n\}$.

Suppose ρ is of the form (1) and $\forall X$ has a negative binding at node v . Then, after substituting ρ for X , we get that in τ node v is marked -1 , while it is marked $+1$ in σ . Thus, by Lemma 7, we cannot get $\tau \sqsubset^* \sigma$. The obtained contradiction proves (ii).

Next, let us suppose that ρ satisfies (2). If this substitution introduces a free variable Z , i.e. $\forall X$ is not in the scope of a quantifier $\forall Z$, then by Lemma 7 (iii), the relation $\tau \sqsubset^* \sigma$ would be impossible. Hence Z must be bound at some node $u \leq w$. Since $\tau \sqsubset^* \sigma$ holds, it follows from Lemma 7 that u must be positive. Hence, by Lemma 8, it follows that $w = u1^m$, for some $m \geq 0$. Let v_1 and v_2 be nodes in σ which are bound by $\forall Z$ and by $\forall X$, respectively. Since $\tau \sqsubset^* \sigma$ holds, it follows that $\forall Z$ is splitable in τ at nodes v_1 and v_2 . Thus, by Lemma 9, we obtain that either $\forall Z$ is dummy in σ or $\forall Z$ and $\forall X$ have only positive bindings in σ . This completes the proof of (iii). ■

Proposition 13 *For arbitrary types σ and τ , if $\sigma \sqsubset \tau$ by a reversible step, then $\vdash \sigma \equiv \tau$.*

Proof: We prove the conclusion assuming first that the reversible step $\sigma \sqsubset \tau$ was performed at a positive node. If this was a (*p-dummy*) step then using (A3) we obtain the conclusion. If this was a (*p-distr*) step then, by Lemma 11, it is enough to use (A4) to obtain the conclusion. Finally, if this step was a (*p-inst*) step, then we apply Lemma 12. In case of (i) or (ii) described in Lemma 12 use (A3). In case of (iii), when $\forall Z$ is dummy (using the notation of this lemma), we use (A2) and (A4) to get the conclusion. In case of (iii), when both $\forall X$ and $\forall Z$ have only positive bindings we use (A3) to get the conclusion.

Now, if the reversible step $\sigma \sqsubset \tau$ was performed at a negative node, then it is easy to check that if X is a new type variable, then the step $(\tau \rightarrow X) \sqsubset (\sigma \rightarrow X)$ is also reversible and is performed at a positive node. Hence, by the first part of this proof we conclude that $\vdash (\tau \rightarrow X) \equiv (\sigma \rightarrow X)$ holds. By Lemma 6 it follows that $\vdash \tau \equiv \sigma$ holds. Thus $\vdash \sigma \equiv \tau$ holds as well. This completes the proof. ■

Proof of Theorem 4: Now we can prove Theorem 4. It follows from Lemma 1, Lemma 2, and Lemma 3, by obvious induction, that if $\vdash \sigma \equiv \tau$ holds, then σ and τ are bicoercible. This proves soundness.

For the completeness part let us assume that σ and τ are bicoercible. Then $\sigma \sqsubset^* \tau$, and all these steps are reversible. Thus, by Proposition 13, we obtain $\vdash \sigma \equiv \tau$. This completes the proof of the theorem.

Acknowledgments: The author would like to thank Pierre-Louis Curien, Roberto Di Cosmo and Giuseppe Longo for stimulating discussions on the topics presented in this paper, during his stay at Ecole Normale Supérieure in 1994, supported by a grant of French Ministry of Research.

References

[Ben93] M. Benke, “Efficient type reconstruction in the presence of inheri-

- tance”, in: A. M. Borzyszkowski, S. Sokolowski (Eds.) *MFCS'93: Mathematical Foundations of Computer science, Proc. 18th Intern. Symp., Gdansk 1993*, Springer-Verlag LNCS **711**, (1993), 272–280.
- [BCGS91] V. Breazu-Tannen, T. Coquand, C.A. Gunter, and A. Scedrov, “Inheritance as implicit coercion”, *Information and Computation* **93**, (1991), 172-221.
- [CMSS94] L. Cardelli, S. Martini, J.C. Mitchell, and A. Scedrov, “An extension of system **F** with subtyping”, *Information and Computation* **94**, (1994), 4-56.
- [CGL92] G. Castagna, G. Ghelli, and G. Longo, “A calculus of overloaded functions with subtyping”, *Proceedings, ACM conference on LISP and Functional Programming*, San Francisco (1992).
- [CP94] G. Castagna, and B.C. Pierce, “Decidable bounded quantification”, *21st Ann. ACM Symposium on Principles of Programming Languages*, (1994), 151-162.
- [Dic95] R. Di Cosmo, *Isomorphisms of types: from λ -calculus to information retrieval and language design*, Birkhauser, 1995.
- [LMS94] G. Longo, K. Milsted, and S. Soloviev, “A logic of subtyping”, Manuscript, Ecole Normale Supérieure, (1994).
- [Mit90] J.C. Mitchell, “Polymorphic type inference and containment”, in: G. Huet (Ed.), *Logical Foundations of Functional Programming*, Addison-Wesley (1990), 153-193.
- [Pie92] B.C. Pierce, “Bounded quantification is undecidable”, *19th Ann. ACM Symposium on Principles of Programming Languages*, (1992), 305-315.
- [Tiu92] J. Tiuryn, “Subtype inequalities”, *Proc. 7-th IEEE Symp. Logic in Computer Science*, Santa Cruz (1992), 308-315.
- [TW93] J. Tiuryn, and M. Wand, “Type reconstruction with recursive types and atomic subtyping”, in: M.-C. Gaudel and J.-P. Jouanaud (Eds.) *TAPSOFT'93: Theory and Practice of Software Development, Proc. 4th Intern. Joint Conf. CAAP/FASE*, Springer-Verlag LNCS **668**, (1993), 686-701.