Predicative Polymorphic Subtyping

Marcin Benke

Institute of Informatics
Warsaw University

June, 1998

Abstract

We consider a version of the Mitchell's polymorphic type containment (aka *subsumption*). Here, it is not suited for the system $\mathcal{F}$ but for the *predicative* version of it introduced by Leivant. The aim of the approach is to propose a new notion of polymorphic functional subsumption which may be decidable unlike Mitchell's one. We define a system for predicative subsumption in the style of Mitchell and then prove its equivalence with a Gentzen-style definition. Next, we study the notion of bicoercibility. This is followed by a reduction of the typability in the Leivant's system with subsumption to subsumption itself.

1 Introduction

The notion of polymorphism formalised as system $\mathcal{F}$ ([Gir72, Rey74]) is widely used in functional programming community. This notion was refined in many different directions in order to obtain a strongly expressible system with user friendly facilities. The ML type system introduced in [Mil78] was a successful milestone in this direction with its decidable type inference [DM82]. On the other hand it came out that system $\mathcal{F}$ itself is less convenient as it has undecidable type inference [Wel94].

People studied different variations of type systems that are closely connected with these two main systems. There was some effort spent in order to enrich the ML system with less restricted quantification. Existential quantification in algebraic datatypes was introduced by Perry in [Per90] and by Läufer and Odersky in [LO94]. This extension is in fact Milner style version of Mitchell and Plotkin abstract types [MP88]. Then the system with existential quantification was extended with universal quantification by [Rém94] which gave a possibility to declare objects with polymorphic methods. Läufer and Odersky introduced in [OL96] a type system with type schemes that may have quantification in any place, but type variables may be instantiated by types with no quantifiers only. All the aforementioned systems have decidable type inference.

Another direction was to change system $\mathcal{F}$ so that one can get decidable type inference. One way consisted in restriction on the form of types. Types of rank at most 2 (rank is the maximal number of steps made to the left on a path leading in a type to a quantifier) lead to decidable type inference while the rank 3 leads to undecidability [KT92]. The idea of changing the system itself was the crucial point in the work of Mitchell [Mit88], where additional relation of *containment* (known also as *subsumption*) was introduced, as well as in the one by Leivant [Lei91], where stratification on type variables was imposed (variable may be substituted for by a type with quantifiers on a lower level).

The Mitchell's system has already its own story. Unfortunately, type inference is undecidable for this system as shown in [Wel96]. Moreover, the containment relation is undecidable, too ([TU96, Wel95]). This was surprising in the light of the fact that the bicoercibility relation (σ is bicoercible with τ iff $\sigma \leq \tau$ and $\tau \leq \sigma$) is decidable [Tiu95].

By contrast the Leivant's system has not been thoroughly studied yet.

Contributions of the paper This paper presents a version of the Mitchell's subsumption relation suited for Leivant's stratified system $\mathcal{F}$. This subsumption relation does not cover the rule for distribution of quantifier, however. This choice was made because proofs are getting very involved in presence of such rule (see [LMS95]). Moreover, it seems that the presence of this rule does not add any substantial complexity as the same mechanism gives undecidability for the Mitchell's system with this rule and without [Chr98]. At last Läufer and Odersky use exactly this subsumption in [OL96].

The notion of subsumption is given in the Mitchell style. We characterise this notion in the spirit of [LMS95] proving its equivalence with a Gentzen-style definition. Next, we study the notion of bicoercibility obtaining very straightforward characterisation of it. A reduction of the typability problem to the problem of deciding subsumption is studied then. The technique is taken from [Jim95]. We end with an algorithm for deriving subsumption in the restricted case for types that have all possible zero-level quantification (variables may be substituted for only by quantifier free types) and may have a vector of level one quantifiers in the root position (variables may be substituted for by terms with zero-level quantification). This result extends the result by Läufer and Odersky [OL96].

Organisation of the paper Section 2 contains preliminary definitions. Section 3 presents the notion of subtyping in the spirit of Mitchell. In Section 4 we introduce the sequent system in the sense of [LMS95] and prove its equivalence with the system defined in Section 3. Section 5 introduces a technical tool of syntax-driven system. The idea of this system is taken from [Wel95]. Bicoercions are studied in Section 6.

2 Definitions

2.1 Stratification

Let t be a natural number. Systems of tier[1] t have the following components:

Type expressions τ and their levels $L(\tau)$ are defined inductively:

- For each level $p = 0, 1, ..., t$ there is a denumerable supply of type variables of level p: $\alpha^p, \alpha^p_1, \ldots$ (we omit the level superscript when it is irrelevant or clear from context). A type variable of level p is also a type expression of level p.
- If σ and τ are type expressions then $\sigma \to \tau$ is a type expression of level $\max(L(\sigma), L(\tau))$
- If τ is a type expression of level p then $\forall\alpha^q.\tau$ is a type expression of level $\max(p, q+1)$

If T is a set of types, then by $L(T)$ we shall mean the multiset of levels

$$\{L(\tau) \mid \tau \in T\}$$

[1] while we use the word *level* to characterize types, the word *tier* is used to denote systems; a sytem of tier t may involve types of levels $0, \ldots, t$.

The relation $\leq_{Mul}$ will denote the usual multiset ordering.

Let τ be a type with $L(\tau) = p+1$ by *rank* of τ we shall mean the maximum length of a path leading in τ to a quantifier binding a variable of level p.

Note that by erasing all level indices from a predicative polymorphic type we get an ordinary "system F" type. We shall denote the result of such erasure performed on type τ by $F(\tau)$.

2.2 Conventions on quantifiers

Following Wells[Wel95, Wel96], we assume the following conventions:

1. Reordering of adjacent quantifiers and α-conversion is allowed at any time.
2. Using α-conversion we assume that no variable is bound more than once in any type and that the names of bound variables are different from the names of free variables.
3. If $\sigma = \forall \alpha^q . \sigma$ and $\alpha^q \notin \mathrm{FV}(\sigma)$ then this binding is called redundant. We consider types differing only with redundant bindings as being equal.

While these conventions may seem arbitrary, we shall see in the subsequent sections that they are indeed justified.

2.3 Representing types: trees and canonical form

Let $\mathcal{P} = \{1,2\}^*$ and $\mathcal{P}^\omega = \{1,2\}^\omega$ be sets of respectively finite and infinite words over $\{1,2\}$ (which may be viewed as paths in binary trees). We shall use capital greek letters (e.g. Π, Δ) to denote such paths.

We may view a type as a tree (with domain being a prefix-closed subset of $\mathcal{P}$) where each arrow corresponds to an internal node, type variables are leaf nodes, and each node may be labelled with a set of quantifiers. We shall denote such a set labelling a node at address Π in type τ by $Q(\tau, \Pi)$. The domain of such a tree shall be called the *tree skeleton* of a type.

2.1 Definition

We say that τ is in the canonical form, if

1. It has no redundant quantifiers.
2. $BV(\tau) = \{\alpha_1, \ldots, \alpha_n\}$ (for some $n \in N$).
3. names of bound variables are different from the names of free variables and no variable is bound more than once.
4. If for some paths Δ, Π where Δ is lexicographically earlier than Π, $\tau_\Delta = \alpha_i$, $\tau_\Pi = \alpha_j$ (and there is no lexicographically earlier path leading to α_j) then $i < j$.
5. At every node the quantifiers are ordered according to indices of the variables they bound.

In [Wel96], Wells introduced another convenient representation for types, which he called *leaf groups*: a type σ is a set of leaf groups $\{G_1, \ldots, G_n\}$. Each leaf group denotes the set of occurrences of a particular variable — free or bound. A group referring to a free variable is of the form $\langle \alpha, P \rangle$

where α is the name of the variable and $P \subseteq \mathcal{P}$ is the set of paths leading to leaves labelled with α; a group referring to a variable bound at the node with address Π is of the form $\langle \Pi, P \rangle$.

The connection between leaf-group representation and canonical forms is established by the following

2.2 Proposition
Two types have the same canonical form if and only if they have the same leaf-group representation.

Both leaf-groups and canonical forms abstract from syntactical details of type representation as well as provide formal foundations for the conventions mentioned in the previous section.

The tree representatiuon allows us to define some useful (partial) functions, used later in the paper:

- $\mathrm{sign}(\Pi)$, defined for $\Pi \in \mathcal{P}$ as $+1$ for positive paths (even number of 1's) and -1 for negative ones.
- $\mathrm{leaf}(\sigma, \Pi)$ — longest prefix of Π leading to a leaf in σ.
- $\mathrm{quant}(\sigma, \Pi)$ — path from root to the quantifier binding the variable at $\mathrm{leaf}(\sigma, \Pi)$, if any.
- $\mathrm{leafvar}(\sigma, \Pi)$ — name of the variable at $\mathrm{leaf}(\sigma, \Pi)$ if it is free, undefined otherwise.
- $\mathrm{quantsign}(\sigma, \Pi) = \mathrm{sign}(\mathrm{quant}(\sigma, \Pi))$ if $\mathrm{quant}(\sigma, \Pi)$ is defined, 0 if the variable is free, undefined if $\mathrm{leaf}(\sigma, \Pi)$ is undefined.

3 Subtyping

3.1 A Hilbert-style system

Axioms:

(refl)	$\sigma \leq \sigma$
(inst)	$\forall \alpha^q . \sigma \leq \forall \vec{\beta}^{\vec{p}}.\ \sigma[\rho/\alpha^q] \quad L(\rho) \leq q \quad \vec{\beta} \notin \mathrm{FV}(\forall \alpha^q . \sigma)$

Rules:

$$(\to)\ \frac{\rho' \leq \rho \quad \tau \leq \tau'}{\rho \to \tau \leq \rho' \to \tau'} \qquad\qquad (\forall)\ \frac{\sigma \leq \sigma'}{\forall \alpha .\, \sigma \leq \forall \alpha .\, \sigma'}$$

$$(\mathrm{trans})\ \frac{\sigma \leq \sigma_1 \quad \sigma_1 \leq \sigma'}{\sigma \leq \sigma'}$$

We shall sometimes use the notation $\sigma \leq_t \tau$ to denote the fact that this inequality is derivable in the system of tier t. We shall also use the notation $\sigma \sqsubset \tau$ to indicate the fact that there is a derivation of $\vdash \sigma \leq \tau$ such that its last rule is other than (trans). It is easy to see that

$$\vdash \sigma \leq \tau \quad \text{iff} \quad \sigma \sqsubset^* \tau$$

3.2 Relationship to the Mitchell's system

Our system differs from the original system of Mitchell [Mit88] (which operates on the types without level annotations) in two points: firstly, the (inst) axiom is restricted to predicative instances. Besides, the system of Mitchell contains additionally an axiom of *distributivity*:

$$\forall\alpha.\,\sigma_1 \to \sigma_2 \leq \forall\alpha.\,\sigma_1 \to \forall\alpha.\,\sigma_2$$

Let $\vdash^M$ denote derivability in the Mitchell's system. We have the following

3.1 Lemma
If $\vdash \sigma \leq \tau$ *then* $\vdash^M F(\sigma) \leq F(\tau)$

4 A sequent system for predicative subtyping

$$\sigma \vdash \sigma$$

$$\frac{\rho' \vdash \rho \quad \tau \vdash \tau'}{\rho \to \tau \vdash \rho' \to \tau'}\ (\to)$$

$$\frac{\sigma[\rho/\alpha^q] \vdash \tau \quad L(\rho) \leq q}{\forall\alpha^q.\sigma \vdash \tau}\ (\forall L)$$

$$\frac{\sigma \vdash \tau}{\sigma \vdash \forall\alpha^q.\tau}\ (\forall R) \qquad \alpha \notin \mathrm{FV}(\sigma)$$

This system differs from [LMS95] in two points: first in the rule $(\forall L)$ instance is predicative, second, the rule $(\forall R)$ is restricted to the case called $(\forall_0\text{-}R)$ in [LMS95]. The latter change reflects the fact that we have no axiom of distributivity in the system defined in the previous section.

4.1 Cut elimination

We shall show that the relation $\vdash$ is transitive, i.e. that the following rule is admissible:

$$\frac{\sigma \vdash \rho \quad \rho \vdash \tau}{\sigma \vdash \tau}\ (cut)$$

Before we do this, let us first note a few properties of the system with (cut) (analogous to some properties proved in [LMS95]):

4.1 Lemma

1. $(\forall R)$ *permutes with* $(\forall L)$*, that is for every derivation ending with the consecutive applications of rules* $(\forall R)$ *and* $(\forall L)$ *there exists a derivation of the same sequent ending with* $(\forall L)$ *followed by* $(\forall R)$*.*
2. $(\forall R)$ *on the right permutes with* (cut)*.*
3. $(\forall L)$ *on the left permutes with* (cut)*.*

Note that our version of the system lacks one permutation property in comparison with [LMS95]: $(\forall R)$ does not permute with $(\rightarrow)$.

Proof:

1. $(\forall R)$ permutes with $(\forall L)$:

$$\dfrac{\dfrac{\sigma \vdash \tau}{\sigma \vdash \forall\alpha.\tau}\ (\forall R)}{\forall\beta.\sigma \vdash \forall\alpha.\tau}\ (\forall L) \quad \leadsto \quad \dfrac{\dfrac{\sigma \vdash \tau}{\forall\beta.\sigma \vdash \tau}\ (\forall L)}{\forall\beta.\sigma \vdash \forall\alpha.\tau}\ (\forall R)$$

2. $(\forall R)$ on the right permutes with (cut):

$$\dfrac{\rho \vdash \sigma \quad \dfrac{\sigma \vdash \tau}{\sigma \vdash \forall\alpha.\tau}\ (\forall R)}{\rho \vdash \forall\alpha.\tau}\ (cut) \quad \leadsto \quad \dfrac{\dfrac{\rho \vdash \sigma \quad \sigma \vdash \tau}{\rho \vdash \tau}\ (cut)}{\rho \vdash \forall\alpha.\tau}\ (\forall R)$$

3. $(\forall L)$ on the left permutes with (cut):

$$\dfrac{\dfrac{\sigma[\rho/\alpha] \vdash \eta}{\forall\alpha.\sigma \vdash \eta}\ (\forall L) \quad \eta \vdash \tau}{\forall\alpha.\sigma \vdash \tau}\ (cut) \quad \leadsto \quad \dfrac{\dfrac{\sigma[\rho/\alpha] \vdash \eta \quad \eta \vdash \tau}{\sigma[\rho/\alpha] \vdash \tau}\ (cut)}{\forall\alpha.\sigma \vdash \tau}\ (\forall L)$$

∎

4.2 Theorem

For any sequent derivable in the system with the rule (cut) there exists an equivalent cut-free derivation

Proof: We shall proceed by induction on the derivation. One-step derivations are obviously cut-free, so what we have to prove is that if in a derivation

$$\dfrac{S_L \quad S_R}{S}\ (cut)$$

the subderivations of S_L and S_R are cut-free then there exists a cut-free derivation of S. We shall do this by analysing how S_L and S_R were derived.

1. The cases where S_L or S_R is an axiom instance are trivial.

2. If S_L was obtained by $(\forall L)$ or S_R was obtained by $(\forall R)$ we may apply the permutation properties established by Lemma 4.1.

3. If S_L was obtained by $(\forall R)$ (and S_R by $(\forall L)$):

$$\dfrac{\dfrac{\sigma \vdash \tau}{\sigma \vdash \forall\alpha.\tau}\ (\forall R) \quad \dfrac{\tau[\pi/\alpha] \vdash \rho}{\forall\alpha.\tau \vdash \rho}}{\sigma \vdash \rho}\ (cut)$$

   Since $\alpha \notin \mathrm{FV}(\sigma)$ (which was a prerequisite of applying $(\forall R)$), by uniformly replacing α by π in the derivation of $\sigma \vdash \tau$ we get a valid (and cut-free) derivation of $\sigma \vdash \tau[\pi/\alpha]$.

   Now, by induction assumption the derivation

$$\dfrac{\sigma \vdash \tau[\pi/\alpha] \quad \tau[\pi/\alpha] \vdash \rho}{\sigma \vdash \rho}\ (cut)$$

   (which has less number of step than the original one) has a cut-free equivalent.

4. S_L was obtained by $(\to)$. Since the case when S_L was obtained by $(\forall R)$ is handled by (2), we may assume that S_R was obtained by $(\to)$ as well:

$$\dfrac{\dfrac{\rho_1 \vdash \sigma_1 \quad \sigma_2 \vdash \rho_2}{\sigma_1 \to \sigma_2 \vdash \rho_1 \to \rho_2}\,(\to) \quad \dfrac{\tau_1 \vdash \rho_1 \quad \rho_2 \vdash \tau_2}{\rho_1 \to \rho_2 \vdash \tau_1 \to \tau_2}\,(\to)}{\sigma_1 \to \sigma_2 \vdash \tau_1 \to \tau_2}\,(cut)$$

we may rearrange the derivation as follows:

$$\dfrac{\dfrac{\tau_1 \vdash \rho_1 \quad \rho_1 \vdash \sigma_1}{\tau_1 \vdash \sigma_1}\,(cut) \quad \dfrac{\sigma_2 \vdash \rho_2 \quad \rho_2 \vdash \tau_2}{\sigma_2 \vdash \tau_2}\,(cut)}{\sigma_1 \to \sigma_2 \vdash \tau_1 \to \tau_2}\,(to)$$

and apply the induction hypothesis.

∎

Cut elimination allows us to state the equivalence of Hilbert- and sequent-style systems:

4.3 Theorem

For any types σ and τ we have

$$\sigma \leq \tau \Leftrightarrow \sigma \vdash \tau$$

Proof: To prove the left-to-right implication we need to show that the rules (corresponding to) $(\forall L)$ and $(\forall R)$ are admissible. We have:

$$\dfrac{\forall\alpha.\sigma \leq \sigma[\rho/\alpha] \quad \sigma[\rho/\alpha] \leq \tau}{\forall\alpha.\sigma \leq \tau}\,(trans)$$

and, if $\alpha \notin \mathrm{FV}(\sigma)$

$$\dfrac{\sigma \leq \forall\alpha.\sigma\tau \quad \dfrac{\sigma \leq \tau}{\forall\alpha.\sigma \leq \forall\alpha.\tau}\,(\forall)}{\sigma \leq \forall\alpha.\tau}\,(trans)$$

On the other hand, the non-trivial facts we need in order to prove the right-to left implication are the following:

1. Admissibility of $(trans)$ — follows from admissibility of (cut).

2. Admissibility of $(\forall)$:

$$\dfrac{\dfrac{\sigma[\alpha/\alpha] \vdash \tau}{\forall\alpha.\,\sigma \vdash \tau}\,(\forall L)}{\forall\alpha.\,\sigma \vdash \forall\alpha.\,\tau}\,(\forall R)$$

3. Derivability of the $(inst)$ axiom: we have to show that $\forall\vec{\alpha}.\,\sigma \vdash \sigma[\vec{\rho}/\vec{\alpha}]$

   (a) $|\vec{\alpha}| = 0$

$$\dfrac{\sigma \vdash \sigma}{\sigma \vdash \forall\gamma_k.\,\sigma}\,(\forall R)$$
$$\vdots$$
$$\dfrac{}{\sigma \vdash \forall\vec{\gamma}.\,\sigma}\,(\forall R)$$

(b) $|\vec{\alpha}| > 0$

$$\cfrac{\cfrac{\sigma[\vec{\rho}/\vec{\alpha}] \vdash \sigma[\vec{\rho}/\vec{\alpha}]}{\forall\alpha_n.\sigma[\rho_1/\alpha_1]\cdots[\rho_{n-1}/\alpha_{n-1}] \vdash \sigma[\vec{\rho}/\vec{\alpha}]}\ (\forall L)}{\vdots}$$

$$\cfrac{\forall\vec{\alpha}.\,\sigma \vdash \sigma[\vec{\rho}/\vec{\alpha}]}{\forall\vec{\alpha}.\,\sigma \vdash \forall\gamma_k.\,\sigma[\vec{\rho}/\vec{\alpha}]}\ (\forall R)$$

$$\vdots$$

$$\cfrac{}{\forall\vec{\alpha}.\,\sigma \vdash \forall\vec{\gamma}.\,\sigma[\vec{\rho}/\vec{\alpha}]}\ (\forall R)$$

∎

5 A syntax-driven system

5.1 Lemma

$\forall\vec{\alpha}.\sigma_1 \to \sigma_2 \leq \forall\vec{\gamma}.\tau_1 \to \tau_2$ (with $\vec{\gamma} \notin \mathrm{FV}(\forall\vec{\alpha}.\sigma_1 \to \sigma_2)$) if and only if there exist $\vec{\rho}$ such that $L(\rho_i) \leq L(\alpha_i)$ as well as $\tau_1 \leq \sigma_1[\vec{\rho}/\vec{\alpha}]$ and $\sigma_2[\vec{\rho}/\vec{\alpha}] \leq \tau_2$

Proof: The right-to-left implication is easy to check:

$$\cfrac{\forall\vec{\alpha}.\sigma_1 \to \sigma_2 \leq \forall\vec{\gamma}.(\sigma_1 \to \sigma_2)[\vec{\rho}/\vec{\alpha}] \quad \cfrac{\cfrac{\tau_1 \leq \sigma_1[\vec{\rho}/\vec{\alpha}] \quad \sigma_2[\vec{\rho}/\vec{\alpha}] \leq \tau_2}{(\sigma_1 \to \sigma_2)[\vec{\rho}/\vec{\alpha}] \leq \tau_1 \to \tau_2}}{\forall\vec{\gamma}.(\sigma_1 \to \sigma_2)[\vec{\rho}/\vec{\alpha}] \leq \forall\vec{\gamma}.\tau_1 \to \tau_2}\ (\forall)}{\forall\vec{\alpha}.\sigma_1 \to \sigma_2 \leq \forall\vec{\gamma}.\tau_1 \to \tau_2}\ (\mathit{trans})$$

On the other hand, by Theorem 4.3 to prove the converse implication it suffices to prove that if

$$\forall\vec{\alpha}.\sigma_1 \to \sigma_2 \vdash \forall\vec{\gamma}.\tau_1 \to \tau_2 \tag{1}$$

then there exist $\vec{\rho}$ such that $\tau_1 \vdash \sigma_1[\vec{\rho}/\vec{\alpha}]$ and $\sigma_2[\vec{\rho}/\vec{\alpha}] \vdash \tau_2$

Let as assume 1. It is easy to see that its derivation must have ended with a sequence of intermixed rules $(\forall L)$ and $(\forall R)$ respectively for α-s and γ-s. However, by Lemma 4.1 all applications of $(\forall R)$ can be moved down. Thus we have

$$\forall\vec{\alpha}.\sigma_1 \to \sigma_2 \vdash \tau_1 \to \tau_2 \tag{2}$$

Now, unless $\vec{\alpha}$ is empty, the only rule yielding such sequent is $(\forall L)$. By applying this argument repeatedly until α is empty, we may conclude that there are $\vec{\rho}$ (of levels compatible with $\vec{\alpha}$) such that

$$(\sigma_1 \to \sigma_2)[\vec{\rho}/\vec{\alpha}] \vdash \tau_1 \to \tau_2$$

from which the thesis follows, since unless this is an axiom instance, the only rule yielding such a sequent is $(\to)$. ∎

The lemma we have just proved leads to the following inference system:

$$\vdash_{sd} \alpha \leq \alpha$$

$$\vdash_{sd} \bot \leq \alpha$$

$$\frac{\vdash_{sd} \tau_1 \leq \sigma_1[\vec{\rho}/\vec{\alpha}] \qquad \vdash_{sd} \sigma_2[\vec{\rho}/\vec{\alpha}] \leq \tau_2}{\vdash_{sd} \forall\vec{\alpha}.\sigma_1 \to \sigma_2 \leq \forall\vec{\gamma}.\tau_1 \to \tau_2}$$

5.2 Theorem
For every types σ, τ we have

$$\vdash \sigma \leq \tau \text{ iff } \vdash_{sd} \sigma \leq \tau$$

Proof: This theorem is in fact a corollary of Lemma 5.1. ∎

5.1 Polarity

5.3 Lemma (Wells)
If $\sigma \leq \tau$ then for all $\Pi \in \mathcal{P}^\omega$ the following properties hold:

1. *If* $\text{quantsign}(\sigma, \Pi) = 0 = \text{quantsign}(\tau, \Pi)$ *then* $\text{leaf}(\sigma, \Pi) = \text{leaf}(\tau, \Pi)$ *and* $\text{leafvar}(\sigma, \Pi) = \text{leafvar}(\tau, \Pi)$.

2. $\text{quantsign}(\sigma, \Pi) \geq \text{quantsign}(\tau, \Pi)$.

3. *If* $\text{leaf}(\sigma, \Pi) < \text{leaf}(\tau, \Pi)$ *then* $\text{quantsign}(\sigma, \Pi) = +1$.

4. *If* $\text{leaf}(\sigma, \Pi) > \text{leaf}(\tau, \Pi)$ *then* $\text{quantsign}(\tau, \Pi) = -1$.

5.4 Lemma
If $\sigma \leq \tau \leq \sigma'$ then for every path $\Pi \in \text{dom}(\sigma) \cap \text{dom}(\sigma')$ we have that $\Pi \in \text{dom}(\tau)$.

Proof: Assume the contrary, i.e. that there is a path Π such that

$$\text{leaf}(\sigma, \Pi) > \text{leaf}(\tau, \Pi) < \text{leaf}(\sigma', \Pi)$$

By Lemma 5.3, since $\sigma \leq \tau$ and $\text{leaf}(\sigma, \Pi) > \text{leaf}(\tau, \Pi)$, we have that $\text{quantsign}(\tau, \Pi) = -1$. On the other hand, since $\tau \leq \sigma'$ and $\text{leaf}(\tau, \Pi) < \text{leaf}(\sigma', \Pi)$, we obtain that $\text{quantsign}(\tau, \Pi) = +1$. Hence, by contradiction, the thesis is proved. ∎

6 Bicoercions

Bicoercion is an important relation introduced by Tiuryn [Tiu95].

Two types σ and τ are said to be *bicoercible* if $\vdash \sigma \leq \tau$ and $\vdash \tau \leq \sigma$ holds.

6.1 Proposition
If σ and τ are bicoercible then $F(\sigma)$ and $F(\tau)$ are bicoercible in $\vdash^M$.

Proof: By Lemma 3.1 $\vdash \sigma \leq \tau$ implies $\vdash^M F(\sigma) \leq F(\tau)$ and $\vdash \tau \leq \sigma$ implies $\vdash^M F(\tau) \leq F(\sigma)$ ∎

In our system, however, bicoercibility is a stronger notion: two types are bicoercible if and only if they have the same canonical form, i.e. are identical up to α conversion, quantifier reordering and redundant quantifiers.

6.1 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) $\forall\alpha\forall\beta.\,\sigma \equiv \forall\beta\forall\alpha.\,\sigma$

(A3) $\forall\alpha.\sigma \equiv \sigma$ if $\alpha \notin FV(\sigma)$

Rules:

$$\frac{\sigma \equiv \sigma' \quad \tau \equiv \tau'}{\sigma \to \sigma' \equiv \tau \to \tau'}\,(\to) \qquad\qquad \frac{\sigma \equiv \sigma'}{\forall\alpha.\,\sigma \equiv \forall\alpha.\,\sigma'}\,(\forall)$$

$$(\text{trans})\ \frac{\sigma \equiv \rho \qquad \rho \equiv \tau}{\sigma \equiv \tau} \qquad\qquad (\text{symm})\ \frac{\sigma \equiv \tau}{\tau \equiv \sigma}$$

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

6.2 Lemma ([Tiu95])
The types $\forall\alpha\beta.\,\sigma$ and $\forall\beta\alpha.\,\sigma$ are bicoercible.

Proof: The proof from [Tiu95] carries over to the predicative case without any changes. ∎

6.3 Lemma
If $\vdash \sigma \equiv \tau$ then σ and τ are bicoercible.

Proof: Soundness of **A2** follows from the previous lemma. Soundness of other axioms and rules is easily verified, since they correspond to respective axioms and rules for subtyping (apart from *symm*, but this rule is obvious anyway). ∎

6.4 Corollary
Every type is bicoercible with its canonical form.

6.2 Properties of bicoercible types

6.5 Lemma
Bicoercible types have identical tree skeletons.

Proof: Assume the contrary, i.e. let σ, τ be bicoercible but with different skeletons. Without loss of generality we may then assume that there is a path Π such that $\Pi \in dom(\sigma) \setminus dom(\tau)$. But since σ and τ are bicoercible, we have that $\sigma \leq \tau \leq \sigma$. Now, it is easily seen that the assumption about Π leads to a contradiction with Lemma 5.4.[2] ∎

[2] In fact this lemma follows from an analogous property of bicoercible types in Mitchell's system, but we choose to present a simple, direct proof.

6.6 Definition
Let σ, τ be types, $\Delta \in \mathcal{P}$. We say that the pair (σ, τ) has a *split* at Δ (or, briefly: a Δ-split) if there is $\Pi \in \mathcal{P}$ such that $\mathrm{quant}(\tau, \Pi) = \Delta$ and $\Pi \notin \mathrm{dom}(\sigma)$. In this case we also say that Π witnesses a Δ-split in (σ, τ).

We shall say that (σ, τ) has an *inverse Δ-split* if (τ, σ) has a Δ-split.

6.7 Lemma
If $\sigma \leq \tau$ then (σ, τ) has no inverse ϵ-splits.

Proof: This follows directly from point 4 of Lemma 5.3. ∎

6.8 Lemma
Let $\sigma = \forall\vec{\alpha}.\sigma', \tau = \forall\vec{\beta}.\tau'$ (where σ', τ' have no quantifiers on top level) be types with no redundant quantifiers and no ϵ-splits. If σ and τ are bicoercible then

$$L(\vec{\alpha}) = L(\vec{\beta})$$

Proof: First we shall prove that

$$L(\vec{\alpha}) \geq_{Mul} L(\vec{\beta})$$

Then the thesis will follow from symmetry of bicoercinbility and the fact that $\geq_{Mul}$ is a partial ordering.

We shall proceed by induction on the derivation of $\vdash \sigma \leq \tau$. If $\sigma \leq \tau$ is an axiom instance then the thesis follows from the absence of ϵ-splits. If the last rule used was $(\rightarrow)$, the thesis is vacuously satisfied, and in case of $(\forall)$ follows from the induction hypothesis.

The interesting case here is when the last rule used was $(trans)$, i.e there is $\rho = \forall\vec{\gamma}.\rho'$ such that $\sigma \leq \rho \leq \tau$. Observe that since σ and τ are bicoercible, we have also $\tau \leq \rho \leq \sigma$. We may assume that ρ has no redundant quantifiers. Moreover, neither (σ, ρ) nor (ρ, τ) have any ϵ-splits since they would be also inverse ϵ-splits (due to bicoercibility). Hence we may apply the induction hypothesis both to (σ, ρ) and to (ρ, τ), obtaining

$$L(\vec{\alpha}) \geq_{Mul} L(\vec{\gamma}) \geq_{Mul} L(\vec{\beta})$$

∎

As the attentive reader most probably observed, Lemma 6.7 ensures that the split condition in the above Lemma is always satisfied for bicoercible types. But in fact, the above lemma can be seen as a corollary from a more general one:

6.9 Lemma
Let $\sigma = \forall\vec{\alpha}.\sigma', \tau = \forall\vec{\beta}.\tau'$ (where σ', τ' have no quantifiers on top level) be types with no redundant quantifiers and no ϵ-splits.If $\sigma \leq \tau$ then

$$L(\vec{\alpha}) \geq_{Mul} L(\vec{\beta})$$

As we only need the weaker version proved above, and the proof is quite involved, we choose not to present it here.

6.10 Theorem
If σ and τ are bicoercible types in canonical form then they are identical.

Proof: If σ and τ are bicoercible then there exist types $\sigma_1, \ldots \sigma_n, \tau_1, \ldots \tau_m$ such that

$$\sigma \sqsubset \sigma_1 \sqsubset \cdots \sigma_n \sqsubset \tau \sqsubset \tau_1 \sqsubset \cdots \sqsubset \tau_m \sqsubset \sigma$$

Such sequence of types will be called a *circle*. We may also assume that all the types are in canonical form. Since all the types in the circle are bicoercible, they all have the same tree skeleton. Now, by inspection of the circle it is easy to prove that if $\forall\vec{\alpha}.\eta, \forall\vec{\beta}.\theta$ are members of the circle (and in canonical form, with η and θ having no quantifiers at the top level) then

1. $L(\vec{\alpha}) = L(\vec{\beta})$ (by Lemma 6.8)
2. hence $\vec{\alpha} = \vec{\beta}$ by the definition of the canonical form
3. hence η and θ are bicoercible

Obviously, if $\sigma_1 \to \sigma_2$ and $\tau_1 \to \tau_2$ are bicoercible then so are σ_1 and τ_1 as well as σ_2 and τ_2. This completes the proof. ∎

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

7 Systems of inequalities

Assume that for each level $p = 0, 1, \ldots, t$ there is a denumerable supply of special type variable symbols of level p: $X^p, X_1^p, \ldots$. The set of special variables occurring in a type σ will be called $SV(\sigma)$.

A system of inequalities

$$\Sigma = \{\sigma_1 \leq \tau_1, \ldots, \sigma_n \leq \tau_n\}$$

is called *satisfiable* if there is a level-respecting substitution S mapping special variables to types such that

$$\vdash S(\sigma_i) \leq S(\tau_i) \qquad 1 \leq i \leq n$$

A system of inequalities Σ is called *orientable* if there exist types σ_L, σ_R such that

1. $SV(\sigma_L) \cup SV(\sigma_R) = SV(\Sigma)$.
2. $SV(\sigma_L) \cap SV(\sigma_R) = \emptyset$.
3. Σ is equivalent to $\{\sigma_L \leq \sigma_R\}$

7.1 Theorem
For every orientable system of inequalities Σ of level k there exist types $\sigma, \tau \in T_{k+1}$ such that Σ is satisfiable iff $\vdash \sigma \leq \tau$.

Proof: Let σ_L, σ_R be such as in definition of orientability and let $\vec{X} = SV(\sigma_L), \vec{Y} = SV(\sigma_R)$. Consider the types

$$\eta_\Sigma = \forall\vec{X}.(\sigma_L \to \bot \to \bot)$$
$$\theta_\Sigma = (\forall\vec{Y}.\sigma_L \to \bot) \to \bot$$

By Lemma 5.1 $\vdash \eta_\Sigma \leq \theta_\Sigma$ iff $\sigma_L \leq \sigma_R$ is satisfiable. ∎

8 Typability

In this section we define a predicative variant of system F with subsumption. Following an idea of Trevor Jim [Jim95, Jim96] typability in this system can be reduced to the satisfiability problem and hence also to the subsumption problem (albeit at the price of increasing level by one).

The set of types is the set of stratified types of level not greater than t defined in section 1.1.

A *type environment* is a function from term variables to types. If $\vec{\alpha}$ is the set of all type variables of level less then t which are free in σ but not in E, then $Gen(E, \sigma) = \forall\vec{\alpha}.\sigma$

The typing judgements $E \vdash M : \sigma$ of predicative system F with subsumption are defined by the following rules:

$$E(x : \sigma) \vdash x : \sigma$$

$$\frac{E(x : \tau) \vdash M : \rho}{E \vdash \lambda x.M : \tau \to \rho}$$

$$\frac{E \vdash M : \tau \to \rho \quad E \vdash N : \tau}{E \vdash MN : \rho}$$

$$\frac{E \vdash M : \tau}{E \vdash M : \sigma} \qquad Gen(E, \sigma) \leq \tau$$

Note that the last rule has as special cases the rules for instantiation and generalisation:

$$\frac{E \vdash M : \tau}{E \vdash M : \forall\alpha.\tau} \qquad \alpha \notin FV(E)$$

$$\frac{E \vdash M : \forall\alpha^q.\tau}{E \vdash M : \tau[\rho/\alpha^q]} \qquad L(\rho) \leq \alpha$$

8.1 Definition
We say that a typing

$$E \vdash M : \sigma$$

is principal for M if (it is a valid typing and) for any other typing $E' \vdash M : \sigma'$ there is a substitution S such that $S(\sigma) \leq \sigma'$ and for all $x \in dom(E)$, we have $E'(x) \leq S(E(x))$.

8.2 Definition
We say that a term M is in *distinct operator form* (in symbols $M \in DOF$) if it is in β-normal form and if a variable x occurs as an operator in M,

$$M = \ldots (xN) \ldots,$$

then x occurs exactly once in M.

8.3 Definition
Associate every term variable x with a distinct type variable α_x and assume that bound and free variables are distinct, and no variable is bound more than once. For any term M in DOF we define the principal pair $PP(M)$ as follows:

1. If $M = \lambda x.N$ and $PP(N) = \langle E, \sigma\rangle$ then
   (a) if $x \in \mathrm{dom}(E)$ then $PP(M) = E_x, E(x) \to \sigma$
   (b) if $x \notin \mathrm{dom}(E)$ then $PP(M) = E, \alpha_x \to \sigma$
2. If $M = xM_1 \dots M_n$ and $PP(M_i) = \langle E_i, \sigma\rangle$ then

8.4 Lemma
If $M \in DOF$, then $PP(M)$ is well defined and if $PP(M) = (E, \sigma)$ then $E \vdash M : \sigma$ is a principal typing for M.

The rule $\beta I(1)$ is the following restriction of βI:

$$(\lambda x.M)N \to_{\beta I(1)} M[N/x] \quad \textit{if } x \textit{ appears exactly once in } M$$

8.5 Lemma
If $M =_{\beta I(1)} N$ then $E \vdash M : \sigma$ iff $E \vdash N : \sigma$

8.6 Definition
Assume we have an infinite supply of fresh variables. For any term M we define M^* as follows:

1. If $M = x$ and y is fresh then $M^* = (yx, y)$.
2. If $M = M_1M_2$, $M_i^* = (M_i', \vec{y}_i)$ then $M^* = (M_1'M_2', \vec{y}_1\vec{y}_2)$.
3. If $M = \lambda x.N$ and $N^* = (N', \vec{y})$ then $M^* = (z(\lambda x.N'), z\vec{y})$.

8.7 Lemma
If $M^* = (M', y_1 \dots y_n)$ then $(\lambda\vec{y}.M') \in DOF$ and

$$(\lambda\vec{y}.M')I \dots I =_{\beta I(1)} M$$

The following theorem, formulated by Jim for system $\mathcal{F}$, carries over to the predficative case:

8.8 Theorem (Jim95)
For a given closed term M there exist types σ, τ such that M is typable if and only if $\sigma \leq \tau$.

9 Future work

We have proven some preliminary results on predicative system $\mathcal{F}$ with subsumption. This work was done in order to make necessary background for studying the type inference problem in this interesting type theory. The basic aim would be to prove decidability in general case — this would give very strong extension of the ML polymorphism. Some partial decidability results going beyond the Läufer and Odersky approach are interesting, too.

References

[Chr98] Jacek Chrząszcz. Polimorphic subtyping without distributivity. MFCS'98 (in this volume), 1998.

[DM82] Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 207–211, 1982.

[Gir72] J.-Y. Girard. *Interprétation fonctionelle et élimination des coupures dans l'arithmetique d'ordre supérieur.* PhD thesis, Université Paris VII, 1972.

[Jim95] Trevor Jim. System F plus subsumption reduces to Mitchell's subtyping relation. Manuscript, 1995.

[Jim96] Trevor Jim. What are principal typings and what are they good for? In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 42–53, 1996.

[KT92] A. J. Kfoury and J. Tiuryn. Type reconstruction in finite-rank fragments of the second-order λ-calculus. *Information and Computation*, 2(98):228–257, June 1992 1992.

[Lei91] Daniel Leivant. Finitely stratified polymorphism. *Information and Computation*, 93:93–113, 1991.

[LMS95] G. Longo, K. Milsted, and S. Soloviev. A logic of subtyping. In *Proc. IEEE Symp. on Logic in Computer Science*, pages 292–299, 1995.

[LO94] Konstantin Läufer and Martin Odersky. Polymorphic type inference and abstract data types. *ACM Transactions on Programming Languages and Systems*, 16(5):1411–1430, September 1994.

[Mil78] Robin Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(14):348–375, December 1978.

[Mit88] John C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76(2/3):211–249, 1988. Reprinted in *Logical Foundations of Functional Programming,* ed. G. Huet, Addison-Wesley (1990) 153–194.

[MP88] John Mitchell and Gordon Plotkin. Abstract types have existential types. *ACM Transactions on Programming Languages and Systems*, 10(3):470–502, 1988.

[OL96] Martin Odersky and Konstantin Läufer. Putting type annotations to work. In *Conf. Rec. ACM Symp. Principles of Programming Languages*, pages 54–67, 1996.

[Per90] N. Perry. *The Implementation of Practical Functional Programming Languages.* PhD thesis, Imperial College of Science, Technology, and Medicine, University of London, 1990.

[Rém94] Didier Rémy. Programming objects with ml-art, and extension to ml with abstract and record types. In *Proceedings of Theoretical Aspects of Programming Languages*, number 789 in LNCS, pages 321–346. Springer, 1994.

[Rey74] J. C. Reynolds. Mathematical foundations of software development. volume 19 of *Lecture Notes in Computer Science*, chapter Towards a theory of type structure., pages 408–425. Springer, 1974.

[Tiu95] Jerzy Tiuryn. Equational axiomatization of bicoercibility for polymorphic types. In Ed. P.S. Thiagarajan, editor, *Proc. 15th Conference Foundations of Software Technology and Theoretical Computer Science*, volume 1026 of *Lecture Notes in Computer Science*, pages 166–179. Springer Verlag, 1995.

[TU96] Jerzy Tiuryn and Paweł Urzyczyn. The subtyping problem for second-order types is undecidable. In *Proceedings of 11th LICS*, 1996.

[Wel94] J. Wells. Typability and type checking in the second-order λ-calculus are equivalent and undecidable. In *Proc. 9th Ann. IEEE Symp. Logic in Comput. Sci.*, pages 176–185, 1994.

[Wel95] J. Wells. The undecidability of Mitchell's subtyping relation. Technical report, Computer Sci. Dept., Boston University, December 1995.

[Wel96] J. B. Wells. Typability is undecidable for F+eta. Technical Report 96-022, Boston University, March 9, 1996.