

The Undecidability of Mitchell's Subtyping Relationship*

J. B. Wells
jbw@cs.bu.edu
Dept. of Computer Science
Boston University
Boston, MA 02215, U.S.A.

December 10, 1995

Abstract

Mitchell defined and axiomatized a *subtyping* relationship (also known as *containment*, *coercibility*, or *subsumption*) over the types of System F (with “ \rightarrow ” and “ \forall ”). This subtyping relationship is quite simple and does not involve bounded quantification. Tiuryn and Urzyczyn quite recently proved this subtyping relationship to be undecidable. This paper supplies a new undecidability proof for this subtyping relationship. First, a new syntax-directed axiomatization of the subtyping relationship is defined. Then, this axiomatization is used to prove a reduction from the undecidable problem of *semi-unification* to subtyping. The undecidability of subtyping implies the undecidability of *type checking* for System F extended with Mitchell's subtyping, also known as “F plus eta”.

1 Introduction

1.1 Background and Motivation

Mitchell originally defined his subtyping relationship, which he called *containment*, to account for a particular meaning of the type constructor “ \rightarrow ” in his semantics for System F. Mitchell devised a notion of a *type inference model* for System F where the meaning of a typing statement “ $M : \tau$ ” is that the meaning of M belongs to a set of λ -term meanings associated with the meaning of τ . This is written as $\llbracket M \rrbracket \in D_{[\tau]}$.

*This work is partly supported by NSF grants CCR-9113196 and CCR-9417382.

In the general case of a type inference model for System F, it is not true that:

$$(1) \quad ([M] \cdot D_{[\tau]}) \subseteq D_{[\rho]} \Rightarrow [M] \in D_{[\tau \rightarrow \rho]}$$

The reverse implication is true:

$$[M] \in D_{[\tau \rightarrow \rho]} \Rightarrow ([M] \cdot D_{[\tau]}) \subseteq D_{[\rho]}$$

This corresponds to the fact that extensionally equal λ -terms can not be given the same types in System F. For example, the λ -term $(\lambda x.Mx)$ where x is fresh can be assigned more types than M .

It is quite natural to consider requiring type inference models to satisfy (1). The stronger semantic restriction on “ \rightarrow ” leads to there being fewer models. Fewer models result in more sound typings that are satisfied by all models. Thus, adding the restriction allows more λ -terms to be typed.

Mitchell describes two ways to make the type system at the syntactic level reflect the addition of the requirement (1) at the semantic level. The first way is to add the eta type inference rule to System F:

$$(\eta) \quad \frac{A \vdash \lambda x.Mx : \sigma \rightarrow \tau}{A \vdash M : \sigma \rightarrow \tau} \quad x \notin \text{FV}(M)$$

This can be seen as requiring extensionally equal λ -terms to be given the same types. The second way is to add the containment rule (more often called *subsumption* by others):

$$(\text{cont}) \quad \frac{A \vdash M : \sigma}{A \vdash M : \tau} \quad \sigma \subseteq \tau$$

where $\sigma \subseteq \tau$ stands for $D_{[\sigma]} \subseteq D_{[\tau]}$ (thus explaining the name “containment”). Mitchell provides an axiomatization (see Section 2.3) that syntactically captures this semantic notion. This axiomatization is what is commonly thought of as Mitchell’s subtyping relationship.

Until quite recently, it has been an open problem whether the subtyping relationship is even decidable. Longo, Milsted, and Soloviev recently devised a new axiomatization of the subtyping relationship (see Section 2.3) which does not contain the (trans) rule:

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

Of course, the relationship is still transitive without this rule. Using this new axiomatization, Tiurny and Urzyczyn recently proved the undecidability of the subtyping relationship by a reduction from the halting problem for 2-counter automata [TU95]. As well as the undecidability of subtyping, Tiurny and Urzyczyn’s result implies the undecidability of *type checking* for System F extended

with subtyping (also known as “F plus eta”). Type checking is the problem where given a set of type assumptions A , a λ -term M , and a type τ , one asks whether $A \vdash M : \tau$ can be derived. There is a trivial reduction from subtyping to type checking where the subtyping question “ $\sigma \subseteq \tau$ ” becomes the type checking question “ $\{x : \sigma\} \vdash x : \tau$ ”.

1.2 Contribution of This Paper

This paper contributes two new results:

1. A new, syntax-directed axiomatization of Mitchell’s subtyping relationship is defined and proven equivalent to the other axiomatizations. The axiomatization of Longo, Milsted, and Soloviev is used as an intermediate step.

This new rule system is syntax-directed in the sense that the syntax of σ uniquely determines the final rule used in proving that σ is a subtype of τ . The syntax-directed nature of the rules provides the important ability to decompose subtypings. A key result of this paper is that the subtyping:

$$\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$$

can be decomposed into two subtypings:

$$\begin{aligned} \tau_L &\subseteq \forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}]) \\ \forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}]) &\subseteq \tau_R \end{aligned}$$

2. The undecidable problem of semi-unification is reduced to the problem of deciding Mitchell’s subtyping relationship. This supplies an alternate proof of the undecidability of subtyping. It also is a good example of how the syntax-directed axiomatization of subtyping makes it easy to prove properties of subtyping.

This undecidability result is also interesting because of the way it proves that type checking for System F extended with subtyping (“F plus eta”) is undecidable. Type checking for System F without subtyping was also proven undecidable by a reduction from semi-unification [Wel94]. Thus, the type checking problem is undecidable for both versions of System F for the same reason.

1.3 Future Work

It has also been an open problem whether *typability* in System F extended with Mitchell’s subtyping is decidable. Using the new syntax-directed axiomatization of subtyping, I have recently discovered a proof of the undecidability of typability in System F extended with subtyping. The overall structure of the

proof is similar to way typability was proven undecidable for System F without subtyping [Wel94], but the lower-level details are quite different. The paper will be available soon.

1.4 Acknowledgements

Jerzy Tiuryn helped me by clearing up some misconceptions I had about the decidability of Mitchell’s subtyping relationship.

Trevor Jim inspired this research by asking me at LICS ’95 to prove the undecidability of typability in System F extended with Mitchell’s subtyping. He also made me aware of Tiuryn’s paper on bicoercibility [Tiu95], which was essential for my understanding of Mitchell’s subtyping relationship.

Assaf Kfoury provided much support and encouragement.

2 Definitions and Foundation

This section introduces basic definitions, notation, and background results by other researchers that are used in this paper.

2.1 General Notation

In general, for any entity X mentioned in this paper, the notation \vec{X}^n denotes the sequence $X_1 X_2 \cdots X_n$. The notation \vec{X} denotes \vec{X}^n for some natural number n that is either unspecified or clear from the context. \vec{X} may also be used to stand for either the set $\{X_1, X_2, \dots, X_n\}$ or the comma-separated sequence X_1, X_2, \dots, X_n , depending on the context.

2.2 Types

The set of types \mathbb{T} is built from the countably infinite set of type variables \mathbb{V} using the “ \rightarrow ” and “ \forall ” type constructors as specified by the grammar

$$\mathbb{T} ::= \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T}) \mid (\forall \mathbb{V}. \mathbb{T})$$

Small Greek letters from the beginning of the alphabet (e.g. $\alpha, \beta, \gamma, \delta$) are metavariables over \mathbb{V} and small Greek letters towards the end of the alphabet (e.g. σ and τ) are metavariables over \mathbb{T} . When writing types, the arrows associate to the right so that $\sigma \rightarrow \tau \rightarrow \rho$ stands for the type $\sigma \rightarrow (\tau \rightarrow \rho)$. The scope of “ \forall .” extends as far to the right as possible. The notation $\forall \vec{\alpha}. \sigma$ stands for $\forall \alpha_1. \cdots \forall \alpha_k. \sigma$, which in turn stands for $\forall \alpha_1. (\cdots (\forall \alpha_k. \sigma) \cdots)$. The symbol \perp is shorthand for $\forall \alpha. \alpha$.

The notation $\sigma[\alpha_1 := \tau_1, \dots, \alpha_n := \tau_n]$ denotes the result of simultaneously substituting τ_i for all free occurrences of α_i in σ , renaming \forall -bound variables in σ as necessary to avoid capturing free variables of τ_i . This may be abbreviated

as $\sigma[\vec{\alpha} := \vec{\tau}]$. For a substitution $S = [\vec{\alpha} := \vec{\sigma}]$, the notation $S(\tau)$ is short for $\tau[\vec{\alpha} := \vec{\sigma}]$, the notation $\text{RAN}(S)$ stands for $\vec{\sigma}$, and the notation $\text{DOM}(S)$ stands for $\vec{\alpha}$. A *renaming* of free type variables is a substitution $[\vec{\alpha} := \vec{\beta}]$ whose range contains only type variables.

The expressions $\text{FTV}(\tau)$ and $\text{BTV}(\tau)$ denote the free and \forall -bound type variables of type τ , respectively. For a set of types \mathbb{X} , the notation $\text{FTV}(\mathbb{X})$ denotes $\bigcup_{\tau \in \mathbb{X}} \text{FTV}(\tau)$.

We have several conventions about how quantifiers in types are treated.

1. Reordering of adjacent quantifiers and α -conversion of types is allowed at any time. For example, we consider the types $\forall\alpha.\forall\beta.\alpha \rightarrow \beta$, $\forall\beta.\forall\alpha.\beta \rightarrow \alpha$, and $\forall\beta.\forall\alpha.\alpha \rightarrow \beta$ to all be equal.
2. Using α -conversion we assume that no variable is \forall -bound more than once in any type, that the \forall -bound type variables of any two type instances are disjoint, and that all \forall -bound type variables of any type instance are disjoint from the free type variables of another type instance.
3. If $\sigma = \forall\alpha.\tau$ and $\alpha \notin \text{FTV}(\tau)$, then “ $\forall\alpha$ ” is a *redundant* quantifier. We do not allow redundant quantifiers to affect the meaning of a type. For example, we consider the types $\forall\beta.\forall\alpha.\alpha$ and $\forall\alpha.\alpha$ to be equal.

We may view a type as a tree where each “ \rightarrow ” corresponds to an internal node, quantifiers are node labels, and other type variable occurrences are leaf nodes. We may refer to particular nodes in the tree by paths from the root where “R” denotes following a right branch and “L” a left branch. Let capital Greek letters (e.g. Π , Δ , Γ) range over these paths. For example, the type β occurs at position LR in the type $(\alpha \rightarrow \beta) \rightarrow \gamma$.

The notion of whether a type’s position within another type is *positive* or *negative* is defined as follows. A type considered as a part of itself occurs positively. If σ occurs positively (respectively negatively) in τ , then it occurs positively (resp. negatively) in both $\rho \rightarrow \tau$ and $\forall\alpha.\tau$ and negatively (resp. positively) in $\tau \rightarrow \rho$. Equivalently, a position is positive if and only if the path to that position contains an even number of occurrences of “L”, otherwise it is negative.

2.3 Subtyping

Mitchell's axiomatization of the subtyping relationship [Mit88] is given by the following rules.

$$\begin{array}{l}
\text{(sub)} \quad \forall \vec{\alpha}. \sigma \subseteq \forall \vec{\beta}. (\sigma[\vec{\alpha} := \vec{\tau}]) \quad \vec{\beta} \notin \text{FTV}(\forall \vec{\alpha}. \sigma) \\
\text{(distr)} \quad \forall \vec{\alpha}. (\sigma \rightarrow \tau) \subseteq (\forall \vec{\alpha}. \sigma) \rightarrow (\forall \vec{\alpha}. \tau) \\
\text{(\(\rightarrow\))} \quad \frac{\sigma_2 \subseteq \sigma_1, \quad \tau_1 \subseteq \tau_2}{\sigma_1 \rightarrow \tau_1 \subseteq \sigma_2 \rightarrow \tau_2} \\
\text{(trans)} \quad \frac{\rho \subseteq \sigma, \quad \sigma \subseteq \tau}{\rho \subseteq \tau} \\
\text{(congruence)} \quad \frac{\sigma \subseteq \tau}{\forall \alpha. \sigma \subseteq \forall \alpha. \tau}
\end{array}$$

Longo, Milsted, and Soloviev give another axiomatization of the subtyping relationship called System F_{co}^{\dagger} [LMS95]. (This name is used because when the inference rules of the system are labelled with proof terms, it becomes a fragment of System F.) It goes as follows.

$$\begin{array}{l}
\text{(ax)} \quad \sigma \vdash_{co} \sigma \\
\text{(\(\rightarrow\))} \quad \frac{\sigma_2 \vdash_{co} \sigma_1, \quad \tau_1 \vdash_{co} \tau_2}{\sigma_1 \rightarrow \tau_1 \vdash_{co} \sigma_2 \rightarrow \tau_2} \\
\text{(\(\forall\)-left)} \quad \frac{\sigma[\alpha := \rho] \vdash_{co} \tau}{\forall \alpha. \sigma \vdash_{co} \tau} \\
\text{(\(\forall_n\)-right)} \quad \frac{\sigma \vdash_{co} \rho_1 \rightarrow \dots \rightarrow \rho_n \rightarrow \tau}{\sigma \vdash_{co} \rho_1 \rightarrow \dots \rightarrow \rho_n \rightarrow (\forall \alpha. \tau)} \quad \alpha \notin \text{FTV}(\{\sigma, \vec{\rho}\})
\end{array}$$

Theorem 11 in [LMS95] states that $\sigma \subseteq \tau$ if and only if $\sigma \vdash_{co} \tau$. Hence, the two systems may be used interchangeably. A type σ is a *subtype* of type τ if and only if there is a derivation in Mitchell's system that $\sigma \subseteq \tau$ and there is a derivation in System F_{co}^{\dagger} that $\sigma \vdash_{co} \tau$.

Definition 2.1 (Sub) The subtyping problem: Given an arbitrary pair of types σ and τ , is it the case that $\sigma \subseteq \tau$?

2.4 Bicoercibility

If σ is a subtype of τ and τ is also a subtype of σ , then σ and τ are *bicoercible*, which we write as $\sigma \equiv \tau$. Tiuryn has proven that the following axiomatization

captures precisely the notion of bicoercibility [Tiu95].

$$(A1) \quad \sigma \equiv \sigma$$

$$(A2) \quad \forall \alpha. \forall \beta. \sigma \equiv \forall \beta. \forall \alpha. \sigma$$

$$(A3) \quad \forall \alpha. \sigma \equiv \sigma[\alpha := \perp] \quad \text{all occurrences of } \alpha \text{ in } \sigma \text{ are positive}$$

$$(A4) \quad \forall \alpha. (\sigma \rightarrow \tau) \equiv \sigma \rightarrow \forall \alpha. \tau \quad \alpha \notin \text{FTV}(\sigma) \text{ and } \alpha \text{ occurs negatively in } \tau$$

$$\text{(arrow)} \quad \frac{\sigma \equiv \sigma', \quad \tau \equiv \tau'}{\sigma \rightarrow \tau \equiv \sigma' \rightarrow \tau'}$$

$$\text{(quant)} \quad \frac{\sigma \equiv \sigma'}{\forall \alpha. \sigma \equiv \forall \alpha. \sigma'}$$

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

$$\text{(symm)} \quad \frac{\sigma \equiv \tau}{\tau \equiv \sigma}$$

Lemma 2.2 *If $\sigma \equiv \tau$, then the following properties hold.*

1. σ and τ have the same tree skeleton.
2. If Π is a leaf in σ and τ , then there is a quantifier for Π in σ if and only if there is one in τ . If there is no quantifier, then the free variable at Π in σ is the same as the free variable at Π in τ . If there is a quantifier for Π , then either it occurs at a positive position in both σ and τ or it occurs at a negative position in both.
3. If Π and Δ are leaves in σ and τ , Π is a positive position, Δ is a negative position, and both Π and Δ are quantified by the same quantifier in σ , then both Π and Δ are quantified by the same quantifier in τ and for any other leaf Γ it is the case that Γ has the same quantifier as Π in σ if and only if this is also the case in τ .

Proof: By inspection of the rules that axiomatize bicoercibility. ■

Lemma 2.3 *If $(\sigma_L \rightarrow \sigma_R) \equiv (\tau_L \rightarrow \tau_R)$ then $\sigma_L \equiv \tau_L$ and $\sigma_R \equiv \tau_R$.*

Proof: This is Lemma 6 in [Tiu95]. ■

2.5 Semi-Unification

For convenience, we define semi-unification using a first-order signature containing the single infix binary function symbol “ \rightarrow ” and for the case where there are only two pairs of terms. (The general definition of semi-unification is reducible to this special case [Pud88, KTU93] and the proof that semi-unification is undecidable is actually for this special case [KTU93].) The set of algebraic terms \mathcal{T} is defined by the grammar $\mathcal{T} ::= \mathbb{V} \mid (\mathcal{T} \rightarrow \mathcal{T})$. This definition is chosen because it allows mapping terms onto types. In fact, $\mathcal{T} \subset \mathbb{T}$.

An *instance* Γ of semi-unification is set of two pairs

$$\Gamma = \{ \tau_1 \dot{\leq} \mu_1, \tau_2 \dot{\leq} \mu_2 \}$$

where $\tau_1, \tau_2, \mu_1, \mu_2 \in \mathcal{T}$. (The use of the symbol “ \leq ” is an established convention. The dot is added in “ $\dot{\leq}$ ” to make it clear that unification is involved.)

An *open substitution* is a function $S: \mathbb{V} \rightarrow \mathcal{T}$ that differs from the identity on only finitely many variables and which extends naturally to a homomorphism $S: \mathcal{T} \rightarrow \mathcal{T}$ so that $S(\sigma \rightarrow \tau) = S(\sigma) \rightarrow S(\tau)$. An open substitution S is a *solution* for an instance Γ of semi-unification if and only if there also exist open substitutions S_1, S_2 such that $S_1(S(\tau_1)) = S(\mu_1)$ and $S_2(S(\tau_2)) = S(\mu_2)$.

Definition 2.4 (SUP) The semi-unification problem: Given an arbitrary instance Γ of semi-unification, does Γ have a solution?

3 Alternate Inference Rules for Subtyping

This section introduces an alternate set of inference rules for Mitchell’s subtyping relationship which have some useful properties which will be used in the next section.

Consider the following new rule system, which we call the *syntax-directed* subtyping rules.

$$\text{(var)} \quad \alpha \leq \alpha$$

$$\text{(\perp)} \quad \perp \leq \sigma$$

$$\text{(\Rightarrow)} \quad \frac{\tau_L \leq \forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}]), \quad \forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}]) \leq \tau_R}{\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \leq \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)}$$

$$\text{where } \vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R) \text{ and } \vec{\gamma} \notin \text{FTV}(\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R))$$

This system is syntax-directed because if $\sigma \leq \tau$ can be derived, then the last rule used in the derivation is uniquely determined by the syntax of σ . The syntax-directed nature of this rule system gives us the following nice property.

Lemma 3.1 *If $\sigma \leq \tau$ and $\sigma = \forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R)$ for some types σ_L and σ_R and some type variables $\vec{\alpha}$, then $\tau = \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$ for some types τ_L and τ_R and some type variables $\vec{\gamma} \notin \text{FTV}(\sigma)$ and there exist some types $\vec{\rho}$ and some type variables $\vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R)$ such that:*

$$\tau_L \leq \forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}]) \quad \text{and} \quad \forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}]) \leq \tau_R$$

Proof: If $\sigma \leq \tau$ then there is a derivation of this fact using only the rules (var), (\perp), and ($\overrightarrow{\Rightarrow}$). If σ contains an “ \rightarrow ”, then the last rule used must have been ($\overrightarrow{\Rightarrow}$). The claim of the lemma is simply the implications of this fact. ■

This new rule system is another axiomatization of Mitchell’s subtyping relationship.

Theorem 3.2 *For all types σ and τ , it holds that $\sigma \leq \tau$ if and only if $\sigma \subseteq \tau$.*

Proof: The two directions of the equivalence are proven separately.

1. To prove that $\sigma \leq \tau$ implies $\sigma \subseteq \tau$, we show that every rule for “ \leq ” is an admissible rule using “ \subseteq ”. We prove each rule separately.
 - (a) (var) This rule is a special case of (sub).
 - (b) (\perp) This rule is a special case of (sub).
 - (c) ($\overrightarrow{\Rightarrow}$) It is given that:

$$\begin{aligned} \tau_L &\subseteq \forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}]) \\ \forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}]) &\subseteq \tau_R \\ \vec{\beta} &\notin \text{FTV}(\sigma_L \rightarrow \sigma_R) \\ \vec{\gamma} &\notin \text{FTV}(\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R)) \end{aligned}$$

and it is desired to show that:

$$\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$$

By (\rightarrow) it is the case that:

$$(\forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}])) \rightarrow (\forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}])) \subseteq \tau_L \rightarrow \tau_R$$

By (distr) it is true that:

$$\forall \vec{\beta}.((\sigma_L[\vec{\alpha} := \vec{\rho}]) \rightarrow (\sigma_R[\vec{\alpha} := \vec{\rho}])) \subseteq (\forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}])) \rightarrow (\forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}]))$$

which is the same as:

$$\forall \vec{\beta}.((\sigma_L \rightarrow \sigma_R)[\vec{\alpha} := \vec{\rho}]) \subseteq (\forall \vec{\beta}.(\sigma_L[\vec{\alpha} := \vec{\rho}])) \rightarrow (\forall \vec{\beta}.(\sigma_R[\vec{\alpha} := \vec{\rho}]))$$

By (sub) and the restriction on $\vec{\beta}$:

$$\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \forall \vec{\beta}.((\sigma_L \rightarrow \sigma_R)[\vec{\alpha} := \vec{\beta}])$$

By three uses of (trans):

$$\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \tau_L \rightarrow \tau_R$$

By (congruence) it is the case that:

$$\forall \vec{\gamma}.\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$$

By (sub) and the restriction on $\vec{\gamma}$:

$$\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \forall \vec{\gamma}.\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R)$$

Then (trans) gives the desired result:

$$\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \subseteq \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$$

2. To prove that $\sigma \subseteq \tau$ implies $\sigma \leq \tau$, we show that every rule for system \vdash_{co} is an admissible rule in system \leq . We prove each rule separately. Remember while reading this that we do not allow the possibility of redundant quantifiers.

- (a) (ax) We want to prove $\sigma \leq \sigma$ for any type σ . The proof goes by induction on the structure of σ .
 - i. ($\sigma = \alpha$) This case is exactly the rule (var).
 - ii. ($\sigma = \rho \rightarrow \tau$) By inductive hypothesis, $\rho \leq \rho$ and $\tau \leq \tau$. Then by rule (\rightarrow) it holds that $(\rho \rightarrow \tau) \leq (\rho \rightarrow \tau)$.
 - iii. ($\sigma = \forall \alpha.\tau$) By cases of whether τ contains an “ \rightarrow ”.
 - A. ($\tau = \alpha$) By (\perp) it holds that $\forall \alpha.\alpha \leq \forall \alpha.\alpha$.
 - B. ($\tau = \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$) By the inductive hypothesis, $\tau_L \leq \tau_L$ and $\tau_R \leq \tau_R$. Thus, we have that:

$$\tau_L \leq \tau_L[\alpha, \vec{\gamma} := \alpha, \vec{\gamma}] \quad \text{and} \quad \tau_R[\alpha, \vec{\gamma} := \alpha, \vec{\gamma}] \leq \tau_R$$

Thus, by (\rightarrow) it holds that:

$$\forall \alpha.\forall \vec{\gamma}.(\tau_L \rightarrow \tau_R) \leq \forall \alpha.\forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$$

which is the desired result.

- (b) (\rightarrow) This is a special case of (\rightarrow).
- (c) (\forall -left) We are given that $\sigma[\alpha := \rho] \leq \tau$ and we wish to show that $\forall \alpha.\sigma \leq \tau$. By cases on whether σ contains an “ \rightarrow ”:

- i. ($\sigma = \alpha$) By (\perp) it holds that $\forall \alpha. \alpha \leq \tau$.
- ii. ($\sigma = \forall \vec{\gamma}. (\sigma_L \rightarrow \sigma_R)$) We assume by α -conversion that $\text{FTV}(\rho) \cap \vec{\gamma} = \emptyset$. Thus, $\sigma[\alpha := \rho] = \forall \vec{\gamma}. ((\sigma_L \rightarrow \sigma_R)[\alpha := \rho])$. It is easy to see that τ contains an “ \rightarrow ”, so let $\tau = \forall \vec{\delta}. (\tau_L \rightarrow \tau_R)$ where $\vec{\delta} \notin \text{FTV}(\forall \vec{\gamma}. ((\sigma_L \rightarrow \sigma_R)[\alpha := \rho]))$. By Lemma 3.1, it must be the case that:

$$\tau_L \leq \forall \vec{\beta}. (\sigma_L[\alpha, \vec{\gamma} := \rho, \vec{\pi}]) \quad \text{and} \quad \forall \vec{\beta}. (\sigma_R[\alpha, \vec{\gamma} := \rho, \vec{\pi}]) \leq \tau_R$$

for some $\vec{\pi}$ and $\vec{\beta} \notin \text{FTV}((\sigma_L \rightarrow \sigma_R)[\alpha := \rho])$. We assume by α -conversion that $\alpha \notin \vec{\beta}$. Thus, $\vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R)$. It is obvious that $\vec{\delta} \notin \text{FTV}(\forall \alpha. \sigma)$. Thus, by rule ($\overline{\rightarrow}$) it holds that:

$$\forall \alpha. \forall \vec{\gamma}. (\sigma_L \rightarrow \sigma_R) \leq \forall \vec{\delta}. (\tau_L \rightarrow \tau_R)$$

which is the desired result.

- (d) (\forall_n -right) We are given that

$$\sigma \leq \rho_1 \rightarrow \cdots \rightarrow \rho_n \rightarrow \tau$$

where $\alpha \notin \text{FTV}(\{\sigma, \vec{\rho}\})$ and we wish to show that

$$\sigma \leq \rho_1 \rightarrow \cdots \rightarrow \rho_n \rightarrow \forall \alpha. \tau$$

By cases on whether σ contains an “ \rightarrow ”.

- i. ($\sigma = \perp$) By (\perp) it holds that $\sigma \leq \rho_1 \rightarrow \cdots \rightarrow \rho_n \rightarrow \forall \alpha. \tau$.
- ii. ($\sigma = \forall \vec{\gamma}. (\sigma_L \rightarrow \sigma_R)$) By induction on n .
- A. ($n = 0$) Let $\tau = \forall \vec{\delta}. (\tau_L \rightarrow \tau_R)$ where $\vec{\delta} \notin \text{FTV}(\sigma)$. It must be the case that

$$\tau_L \leq \forall \vec{\beta}. (\sigma_L[\vec{\gamma} := \vec{\rho}]) \quad \text{and} \quad \forall \vec{\beta}. (\sigma_R[\vec{\gamma} := \vec{\rho}]) \leq \tau_R$$

for some $\vec{\rho}$ and $\vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R)$. Thus, by ($\overline{\rightarrow}$) we have that $\forall \vec{\gamma}. (\sigma_L \rightarrow \sigma_R) \leq \forall \alpha. \forall \vec{\delta}. (\tau_L \rightarrow \tau_R)$ which is exactly $\sigma \leq \forall \alpha. \tau$, the desired result.

- B. ($n > 0$) It must be the case that

$$\begin{aligned} \rho_1 &\leq \forall \vec{\beta}. (\sigma_L[\vec{\gamma} := \vec{\rho}]) \\ \forall \vec{\beta}. (\sigma_R[\vec{\gamma} := \vec{\rho}]) &\leq \rho_2 \rightarrow \cdots \rightarrow \rho_n \rightarrow \tau \end{aligned}$$

for some $\vec{\rho}$ and $\vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R)$. By cases depending on whether $\alpha \in \text{FTV}(\forall \vec{\beta}. (\sigma_R[\vec{\gamma} := \vec{\rho}]))$.

- Suppose it is the case that $\alpha \in \text{FTV}(\forall \vec{\beta}.(\sigma_R[\vec{\gamma} := \vec{\rho}]))$. Since $\alpha \notin \text{FTV}(\sigma)$, it must be that α appears in one of the $\vec{\rho}$. Observe that we have already shown (\forall -left) and (\forall_0 -right) to be admissible for “ \leq ”. By (\forall -left) it must be that:

$$\forall \alpha. \forall \vec{\beta}. (\sigma_R[\vec{\gamma} := \vec{\rho}]) \leq \rho_2 \rightarrow \dots \rightarrow \rho_n \rightarrow \tau$$

By the induction hypothesis (for $n - 1$) it must be true that:

$$\forall \alpha. \forall \vec{\beta}. (\sigma_R[\vec{\gamma} := \vec{\rho}]) \leq \rho_2 \rightarrow \dots \rightarrow \rho_n \rightarrow \forall \alpha \tau$$

By (\forall_0 -right) it is the case that $\rho_1 \leq \forall \alpha. \forall \vec{\beta}. (\sigma_L[\vec{\gamma} := \vec{\rho}])$. By ($\overline{\rightarrow}$) we have:

$$\forall \vec{\gamma}. (\sigma_L \rightarrow \sigma_R) \leq \rho_1 \rightarrow \dots \rightarrow \rho_n \rightarrow \forall \alpha \tau$$

which is the desired result.

- Suppose that $\alpha \notin \text{FTV}(\forall \vec{\beta}.(\sigma_R[\vec{\gamma} := \vec{\rho}]))$. Then by induction hypothesis:

$$\forall \vec{\beta}. (\sigma_R[\vec{\gamma} := \vec{\rho}]) \leq \rho_2 \rightarrow \dots \rightarrow \rho_n \rightarrow \forall \alpha \tau$$

Then ($\overline{\rightarrow}$) gives the desired result:

$$\forall \vec{\gamma}. (\sigma_L \rightarrow \sigma_R) \leq \rho_1 \rightarrow \dots \rightarrow \rho_n \rightarrow \forall \alpha \tau$$

■

Now that we have proven that Mitchell’s subtyping rules are equivalent to the syntax-directed subtyping rules, we may use them interchangeably.

4 Reducing Semi-Unification to Subtyping

This section contains the reduction from semi-unification to Mitchell’s subtyping relationship, which proves the undecidability of the latter.

Theorem 4.1 *SUP is reducible to SUB. Specifically, given an arbitrary instance of SUP:*

$$\Gamma = \{ \sigma_1 \dot{\leq} \tau_1, \sigma_2 \dot{\leq} \tau_2 \}$$

we can construct a pair of types μ and π such that $\mu \leq \pi$ if and only if Γ has a solution.

Proof: Let $\vec{\gamma} = \text{FTV}(\{\sigma_1, \sigma_2, \tau_1, \tau_2\})$. With no loss of generality, assume that $\text{FTV}(\{\sigma_1, \sigma_2\}) \supseteq \text{FTV}(\{\tau_1, \tau_2\})$. Let $\vec{\rho}$ stand for $\rho \rightarrow \rho$. Define μ and π as follows:

$$\begin{aligned} \mu &= \forall \beta. ((\beta \rightarrow ((\beta \rightarrow \beta \rightarrow \perp) \rightarrow \perp) \rightarrow \beta) \rightarrow \perp) \\ \pi &= \pi_1 \rightarrow \perp \\ \pi_1 &= \forall \vec{\alpha}. \forall \vec{\gamma}. (\quad (\overline{(\sigma_1 \rightarrow \sigma_2)} \rightarrow \perp) \\ &\quad \rightarrow (\quad (\overline{(\tau_1 \rightarrow \alpha_1)} \rightarrow \perp) \\ &\quad \quad \rightarrow (\overline{(\alpha_2 \rightarrow \tau_2)} \rightarrow \perp) \\ &\quad \quad \rightarrow \perp) \\ &\quad \rightarrow \perp) \\ &\quad \rightarrow (\overline{(\sigma_1 \rightarrow \sigma_2)} \rightarrow \perp)) \end{aligned}$$

We now prove that there is a solution for Γ if and only if $\mu \leq \pi$.

First, we prove that $\mu \leq \pi$ is equivalent to the existence of several substitutions that satisfy a set of four other subtypings. We do this by deconstructing $\mu \leq \pi$ using repeated applications of Lemma 3.1.

Since μ contains an “ \rightarrow ”, one use of Lemma 3.1 shows that $\mu \leq \pi$ if and only if there exist a substitution T_1 whose domain is $\{\beta\}$ and a set of variables $\vec{\delta}_1$ drawn only from the fresh variables in $\text{FTV}(\text{RAN}(T_1))$ such that:

$$(2) \quad \begin{aligned} \pi_1 &\leq \forall \vec{\delta}_1. T_1(\beta \rightarrow ((\beta \rightarrow \beta \rightarrow \perp) \rightarrow \perp) \rightarrow \beta) \\ \perp &\leq \perp \end{aligned}$$

The astute reader will notice that $\perp \leq \perp$ is always true. From now on, whenever an application of Lemma 3.1 generates $\perp \leq \perp$ as one of the two subtypings that must be true, we will ignore it. Lemma 3.1 again tells us that subtyping (2) holds if and only if there exist a substitution T_2 whose domain is $\vec{\alpha} \cup \vec{\gamma}$ and a set of variables $\vec{\delta}_2$ drawn only from the fresh variables in $\text{FTV}(\text{RAN}(T_2))$ such that:

$$(3) \quad T_1(\beta) \leq \forall \vec{\delta}_2. T_2(\overline{(\sigma_1 \rightarrow \sigma_2)} \rightarrow \perp)$$

and also:

$$(4) \quad \begin{aligned} \forall \vec{\delta}_2. T_2(\quad (\quad (\overline{(\tau_1 \rightarrow \alpha_1)} \rightarrow \perp) \\ \quad \rightarrow (\overline{(\alpha_2 \rightarrow \tau_2)} \rightarrow \perp) \\ \quad \rightarrow \perp) \\ \quad \rightarrow \perp) \\ \rightarrow (\overline{(\sigma_1 \rightarrow \sigma_2)} \rightarrow \perp)) \\ \leq T_1(((\beta \rightarrow \beta \rightarrow \perp) \rightarrow \perp) \rightarrow \beta) \end{aligned}$$

Lemma 3.1 gives the result that subtyping (4) holds if and only if there exist a substitution T_3 whose domain is $\vec{\delta}_2$ and a set of variables $\vec{\delta}_3$ drawn from the

fresh variables in $\text{FTV}(\text{RAN}(T_3))$ such that the following subtypings hold:

$$(5) \quad T_1((\beta \rightarrow \beta \rightarrow \perp) \rightarrow \perp) \leq \forall \vec{\delta}_3. T_3(T_2(\quad (\quad ((\overline{\tau_1 \rightarrow \alpha_1}) \rightarrow \perp) \\ \rightarrow ((\overline{\alpha_2 \rightarrow \tau_2}) \rightarrow \perp) \\ \rightarrow \perp) \\ \rightarrow \perp))$$

$$(6) \quad \forall \vec{\delta}_3. T_3(T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp)) \leq T_1(\beta)$$

From (5) by Lemma 3.1 we get this equivalent subtyping:

$$(7) \quad T_3(T_2(((\overline{\tau_1 \rightarrow \alpha_1}) \rightarrow \perp) \rightarrow ((\overline{\alpha_2 \rightarrow \tau_2}) \rightarrow \perp) \rightarrow \perp)) \leq T_1(\beta \rightarrow \beta \rightarrow \perp)$$

Lemma 3.1 gives us these subtypings that are equivalent to subtyping (7):

$$(8) \quad T_1(\beta) \leq T_3(T_2((\overline{\tau_1 \rightarrow \alpha_1}) \rightarrow \perp))$$

$$(9) \quad T_3(T_2(((\overline{\alpha_2 \rightarrow \tau_2}) \rightarrow \perp) \rightarrow \perp)) \leq T_1(\beta \rightarrow \perp)$$

From (9) by Lemma 3.1 we get this subtyping:

$$(10) \quad T_1(\beta) \leq T_3(T_2((\overline{\alpha_2 \rightarrow \tau_2}) \rightarrow \perp))$$

Thus, by repeated use of Lemma 3.1 we have learned that $\mu \leq \pi$ if and only if there exist three substitutions T_1 , T_2 , and T_3 and three sets of variables $\vec{\delta}_1$, $\vec{\delta}_2$, and $\vec{\delta}_3$ such that these subtypings are true:

$$(3) \quad T_1(\beta) \leq \forall \vec{\delta}_2. T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp)$$

$$(6) \quad \forall \vec{\delta}_3. T_3(T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp)) \leq T_1(\beta)$$

$$(8) \quad T_1(\beta) \leq T_3(T_2((\overline{\tau_1 \rightarrow \alpha_1}) \rightarrow \perp))$$

$$(10) \quad T_1(\beta) \leq T_3(T_2((\overline{\alpha_2 \rightarrow \tau_2}) \rightarrow \perp))$$

and these constraints are satisfied:

$$\text{DOM}(T_1) = \{\beta\}$$

$$\text{DOM}(T_2) = \vec{\alpha} \cup \vec{\gamma}$$

$$\vec{\delta}_2 \in \text{FTV}(\text{RAN}(T_2)) - (\vec{\alpha} \cup \vec{\gamma})$$

$$\text{DOM}(T_3) = \vec{\delta}_2$$

$$\vec{\delta}_3 \in \text{FTV}(\text{RAN}(T_3)) - (\text{FTV}(\text{RAN}(T_2)) - \vec{\delta}_2)$$

Now we show that these subtypings and constraints are satisfied if and only if Γ has a solution. The two directions of the equivalence are proven separately.

1. Suppose Γ has a solution. In other words, there are open substitutions S , S_1 , and S_2 such that:

$$S_1(S(\sigma_1)) = S(\tau_1) \quad \text{and} \quad S_2(S(\sigma_2)) = S(\tau_2)$$

Pick $T_1, T_2, T_3, \vec{\delta}_1, \vec{\delta}_2,$ and $\vec{\delta}_3$ so that the following equations are true:

$$\begin{aligned}
T_2(\alpha_1) &= S_1(S(\sigma_2)) \\
T_2(\alpha_2) &= S_2(S(\sigma_1)) \\
T_2(\gamma_i) &= S(\gamma_i) \\
\vec{\delta}_1 &= \emptyset \\
\vec{\delta}_2 &= \text{FTV}(\text{RAN}(S)) \\
T_1(\beta) &= \forall \vec{\delta}_2 . S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \\
T_3(\alpha) &= \alpha' \quad (\text{i.e. a renaming}) \\
\vec{\delta}_3 &= T_3(\vec{\delta}_2)
\end{aligned}$$

Then the subtypings (3), (6), (8), and (10) become the following:

$$\begin{aligned}
(11) \quad & \forall \vec{\delta}_2 . S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \leq \forall \vec{\delta}_2 . S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \\
(12) \quad & \forall \vec{\delta}_3 . T_3(S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp)) \leq \forall \vec{\delta}_2 . S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \\
(13) \quad & \forall \vec{\delta}_2 . S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \leq T_3((\overline{S(\tau_1) \rightarrow S_1(S(\sigma_2))}) \rightarrow \perp) \\
(14) \quad & \forall \vec{\delta}_2 . S((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \leq T_3((\overline{S_2(S(\sigma_1)) \rightarrow S(\tau_2)}) \rightarrow \perp)
\end{aligned}$$

It is easy to check that (11), (12), (13), and (14) are true.

2. Suppose $\mu \leq \pi$ is true. Then the subtypings (3), (6), (8), and (10) and their associated constraints must be true.

By the (sub) rule, the following subtyping holds:

$$(15) \quad \forall \vec{\delta}_2 . T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \leq \forall \vec{\delta}_3 . T_3(T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp))$$

By (15), (3), and (6), the following bicoercibility holds:

$$(16) \quad T_1(\beta) \equiv \forall \vec{\delta}_2 . T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \equiv \forall \vec{\delta}_3 . T_3(T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp))$$

Let T_4 be the result of restricting the domain of T_3 to $\text{FTV}(T_2(\vec{\gamma}))$. Thus, it is the case that:

$$\forall \vec{\delta}_2 . T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) \equiv \forall \vec{\delta}_3 . T_4(T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp))$$

If for some $\delta_{2,i}$ it were the case that $T_4(\delta_{2,i})$ contained an “ \rightarrow ”, then this would mean that two bicoercible types had different tree skeletons, contradicting property 1 of Lemma 2.2. Thus, every member of $\text{RAN}(T_4)$ is either \perp or a type variable. Now observe that every free variable in $T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp)$ occurs in at least one positive position and at least one negative position. If for some $\delta_{2,i}$ it were the case that $T_4(\delta_{2,i}) = \perp$, then there would be a negative occurrence of $\delta_{2,i}$ at Π such that the quantifier

for Π in $\forall \vec{\delta}_2. T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp)$ was at a positive position (the root) but the quantifier for Π in $\forall \vec{\delta}_3. T_4(T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp))$ was at a negative position (at Π). This would contradict property 2 of Lemma 2.2. Thus, every member of $\text{RAN}(T_4)$ is a type variable. If for $i \neq j$ it were the case that $T_4(\delta_{2,i}) = T_4(\delta_{2,j})$, then this would contradict property 3 of Lemma 2.2. Thus, T_4 is a renaming of type variables.

By (16), (8), and (10), the following subtypings must hold:

$$\begin{aligned} \forall \vec{\delta}_2. T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) &\leq T_3(T_2((\overline{\tau_1 \rightarrow \alpha_1}) \rightarrow \perp)) \\ \forall \vec{\delta}_2. T_2((\overline{\sigma_1 \rightarrow \sigma_2}) \rightarrow \perp) &\leq T_3(T_2((\overline{\alpha_2 \rightarrow \tau_2}) \rightarrow \perp)) \end{aligned}$$

By Lemma 3.1, there must exist substitutions T_5 and T_6 with the appropriate domains and sets of variables $\vec{\delta}_5$ and $\vec{\delta}_6$ satisfying the appropriate constraints such that these subtypings must then be true:

$$\begin{aligned} T_3(T_2(\overline{\tau_1 \rightarrow \alpha_1})) &\leq \forall \vec{\delta}_5. T_5(T_2(\overline{\sigma_1 \rightarrow \sigma_2})) \\ T_3(T_2(\overline{\alpha_2 \rightarrow \tau_2})) &\leq \forall \vec{\delta}_6. T_6(T_2(\overline{\sigma_1 \rightarrow \sigma_2})) \end{aligned}$$

This is shorthand for these subtypings:

$$\begin{aligned} T_3(T_2((\tau_1 \rightarrow \alpha_1) \rightarrow (\tau_1 \rightarrow \alpha_1))) &\leq \forall \vec{\delta}_5. T_5(T_2((\sigma_1 \rightarrow \sigma_2) \rightarrow (\sigma_1 \rightarrow \sigma_2))) \\ T_3(T_2((\alpha_2 \rightarrow \tau_2) \rightarrow (\alpha_2 \rightarrow \tau_2))) &\leq \forall \vec{\delta}_6. T_6(T_2((\sigma_1 \rightarrow \sigma_2) \rightarrow (\sigma_1 \rightarrow \sigma_2))) \end{aligned}$$

By Lemma 3.1, we now get four subtypings which can actually be written as the following two bicoercibilities:

$$\begin{aligned} T_5(T_2(\sigma_1 \rightarrow \sigma_2)) &\equiv T_3(T_2(\tau_1 \rightarrow \alpha_1)) \\ T_6(T_2(\sigma_1 \rightarrow \sigma_2)) &\equiv T_3(T_2(\alpha_2 \rightarrow \tau_2)) \end{aligned}$$

By Lemma 2.3, this is equivalent to these four bicoercibilities:

$$\begin{array}{ll} T_5(T_2(\sigma_1)) \equiv T_3(T_2(\tau_1)) & T_5(T_2(\sigma_2)) \equiv T_3(T_2(\alpha_1)) \\ T_6(T_2(\sigma_2)) \equiv T_3(T_2(\tau_2)) & T_6(T_2(\sigma_1)) \equiv T_3(T_2(\alpha_2)) \end{array}$$

We will ignore the bicoercibilities in the right column. The left pair are equivalent to these, with T_4 used instead of T_3 :

$$\begin{aligned} T_5(T_2(\sigma_1)) &\equiv T_4(T_2(\tau_1)) \\ T_6(T_2(\sigma_2)) &\equiv T_4(T_2(\tau_2)) \end{aligned}$$

Since T_4 is a renaming of type variables, it is reasonable to consider its inverse, T_4^{-1} . Clearly, the following equations must be true:

$$\begin{aligned} T_4^{-1}(T_5(T_2(\sigma_1))) &\equiv T_2(\tau_1) \\ T_4^{-1}(T_6(T_2(\sigma_2))) &\equiv T_2(\tau_2) \end{aligned}$$

At this point, the existence of a solution for Γ has almost been shown. The only problem can be if T_2 , T_5 , and T_6 mention quantifiers in their ranges. It is sufficient to erase these quantifiers to produce a solution for Γ . Pick a type variable ε such that $\varepsilon \notin \text{DOM}(T_5) \cup \text{DOM}(T_6) \cup \text{RAN}(T_4)$. Define E to erase quantifiers as follows:

$$\begin{aligned} E(\alpha) &= \alpha \\ E(\sigma \rightarrow \tau) &= E(\sigma) \rightarrow E(\tau) \\ E(\forall \alpha. \sigma) &= E(\sigma[\alpha := \varepsilon]) \end{aligned}$$

It is the case that if $\sigma \equiv \tau$, then $E(\sigma) = E(\tau)$. (This is because σ and τ can only differ in the positions of quantifiers and the names of bound variables.) Thus, it follows that:

$$\begin{aligned} E(T_4^{-1}(T_5(T_2(\sigma_1)))) &= E(T_2(\tau_1)) \\ E(T_4^{-1}(T_6(T_2(\sigma_2)))) &= E(T_2(\tau_2)) \end{aligned}$$

Define open substitutions S , S_1 , and S_2 as follows:

$$\begin{aligned} S &= T_2 \circ E \\ S_1 &= T_5 \circ T_4^{-1} \circ E \\ S_2 &= T_6 \circ T_4^{-1} \circ E \end{aligned}$$

Thus, we know that:

$$\begin{aligned} S_1(T_2(\sigma_1)) &= S(\tau_1) \\ S_2(T_2(\sigma_2)) &= S(\tau_2) \end{aligned}$$

It is easy to see that for any substitution T where $\varepsilon \notin \text{DOM}(T)$ that $T \circ E = E \circ T \circ E$. Thus, $T_2 \circ S_1 = S \circ S_1$ and $T_2 \circ S_2 = S \circ S_2$. This gives the desired result:

$$\begin{aligned} S_1(S(\sigma_1)) &= S(\tau_1) \\ S_2(S(\sigma_2)) &= S(\tau_2) \end{aligned}$$

■

Theorem 4.2 *SUB is undecidable, i.e. it is undecidable whether $\sigma \leq \tau$ for arbitrary σ and τ .*

Proof: By Theorem 4.1, SUP is reducible to SUB. SUP was proven to be undecidable by Kfoury, Tiuryn, and Urzyczyn [KTU93]. ■

References

- [KTU93] A. J. Kfoury, J. Tiuryn, and P. Urzyczyn. The undecidability of the semi-unification problem. *Inf. Comput.*, 102(1):83–101, Jan. 1993.
- [LMS95] G. Longo, K. Milsted, and S. Soloviev. A logic of subtyping. In *Proc. 10th Ann. IEEE Symp. Logic Comput. Sci.*, pp. 292–299, June 26–29, 1995.
- [Mit88] J. C. Mitchell. Polymorphic type inference and containment. *Inf. Comput.*, 76(2/3):211–249, Feb./Mar. 1988.
- [Pud88] P. Pudlák. On a unification problem related to Kreisel’s conjecture. *Commentationes Mathematicae Universitatis Carolinae*, 29(3):551–556, 1988. Prague, Czechoslovakia.
- [Tiu95] J. Tiuryn. Equational axiomatization of bicoercibility for polymorphic types. Technical Report 95-004, Comp. Sci. Dept., Boston Univ., Feb. 1995. URL: <ftp://cs-ftp.bu.edu/techreports/95-004-coercibility.ps.Z>.
- [TU95] J. Tiuryn and P. Urzyczyn. The subtyping problem for second-order types is undecidable. Technical report, Inst. of Informatics, Univ. of Warsaw, Nov. 1995. URL: <ftp://ftp.mimuw.edu.pl/pub/users/urzy/sub-undec.ps.Z>.
- [Wel94] J. B. Wells. Typability and type checking in the second-order λ -calculus are equivalent and undecidable. In *Proc. 9th Ann. IEEE Symp. Logic Comput. Sci.*, July 4–6, 1994.