ECOLE NORMALE SUPERIEURE

Coherence and Transitivity of Subtyping as Entailment

Giuseppe LONGO
Kathleen MILSTED
Sergei SOLOVIEV

LIENS - 96 - 18

Département de Mathématiques et Informatique

CNRS URA 1327

Coherence and Transitivity of Subtyping as Entailment

Giuseppe LONGO
Kathleen MILSTED[*]
Sergei SOLOVIEV[**]

LIENS - 96 - 18

October 1996

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00
Adresse électronique : longo@dmi.ens.fr

[*]France Télécom CNET, 38-40 rue G. Leclerc
92131 Issy-les-Moulineaux

Adresse électronique :milsted@issy.cnet.fr

[**]Durham Uvty, Comp Sci Dept, Science Labs
South Road, Durham DH1 3LE, Great Britain

Adresse électronique : sergei.soloviev@durham.ac.uk

Coherence and Transitivity of Subtyping as Entailment[*]

Giuseppe Longo
LIENS(CNRS) and DMI
Ecole Normale Supérieure
45 rue d'Ulm
75005 Paris, France
longo@dmi.ens.fr

Kathleen Milsted
France Télécom CNET
38–40 rue du Gen. Leclerc
92131 Issy-les-Moulineaux
France
milsted@issy.cnet.fr

Sergei Soloviev
Durham University
Comp Sci Dept, Science Labs
South Road, Durham DH1 3LE
Great Britain
sergei.soloviev@durham.ac.uk

September 1996

Abstract

The relation of *inclusion* between types has been suggested by the practice of programming as it enriches the polymorphism of functional languages. We propose a simple (and linear) sequent calculus for subtyping as logical entailment. This allows us to derive a complete and coherent approach to subtyping from a few, logically meaningful sequents. In particular, transitivity and anti-symmetry will be derived from elementary logical principles.

1 Introduction

1.1 Motivations, theories and models

In recent years, several extensions of core functional languages have been proposed to deal with the notion of subtyping; see, for example, [CW85, Mit88, BL90, BCGS91, CMMS91, CG92, PS94, Tiu96, TU96]. These extensions were suggested by the practice of programming in computer science. In particular, they were inspired by the notion of inheritance as used in object-oriented programming languages, or by other concrete implementations of the following form of polymorphism: data living in a type σ, which is a subtype of τ, may also be seen as living in type τ, in some suitable sense. So, an integer is also a real, modulo an obvious "almost identical" coercion from integers to reals.

However, subtyping, in the presence of the functional arrow $\rightarrow$ (and second order universal quantification $\forall$) presents some problems. Indeed, in all functional approaches

[*]This paper is a completely revised version of the paper "A Logic of Subtyping"; a preliminary version of the latter paper, with no proofs and not dealing with base types, appeared in the proceedings of the LICS'95 Symposium (San Diego, U.S.A.), July 1995, and a full version is available by anonymous ftp from ftp.ens.fr as /pub/dmi/users/longo/logicSubtyping.ps.Z.

to subtyping, arrow is formalized as being contravariant or antimonotone in the first argument. More formally:

$$(\rightarrow) \qquad \frac{\sigma \leq \tau \qquad \rho \leq \rho'}{\tau \rightarrow \rho \ \leq \ \sigma \rightarrow \rho'}$$

where $\sigma \leq \tau$ is read "σ is a subtype of τ". The contravariant behavior of $\rightarrow$ (on the left) intuitively fits with the categorical notion of (contravariant) Hom functor, but also with the intuitive understanding of programs as transformations acting on inputs: if a program M acts on inputs N in τ, then it can take as input any element in a subtype σ of τ. This poses a first well-known mathematical challenge: is it possible to give a general mathematical meaning of this formal construct and universal quantification, in the sense, say, of denotational semantics or of logical calculi?

1.1.1 Proof-theoretic analyses

In light of the proof-theoretic investigations that the problem of subtyping has stimulated, it is fair to say that "the notion of subtyping is one of the most important concepts introduced recently into the theory of functional languages" [Tiu96].

Let's quote some relevant papers. [CG92] solves the difficult problems of coherence and minimum typing. Clearly, if a term belongs to a type and to any larger type, then it has no unique type nor does it code a unique proof (or type derivation). Of course, the contravariant behavior of $\rightarrow$ (on the left) and second order quantification complicate the problem. Yet, in [CG92], it is shown that each proof reduces to a unique "normal" one, which also yields the minimum type of its coding term. Other extensions of various lambda-calculi further clarified the issue of subtyping at a syntactic level. The approach in [BCGS91] significantly departs from the "intended" meaning in the previous papers: the subtyping relation is interpreted by the existence of a certain definable term between type expressions. These terms are called "*coercions*".

Yet another approach may be found in [CMMS91]. This is directly related to Cardelli's ideas for the programming language Quest and contains its main features as a basis (the Top type, bounded quantification, etc.). In short, a type-inference system is proposed which fully formalizes Quest's rules and investigates conservativity of typing judgements and some categorical properties in a proof-theoretic frame (e.g., the syntactic isomorphisms between closed terms). Moreover, [CMMS91] suggests a rule for equality of terms, (eq appl2), a variant of which will be largely used in our approach.

Both [BCGS91] and [CMMS91] are "orthogonal" to this paper, as they contain features (a Top type, records, variants, bounded quantification, etc.) that were motivated mostly by the practice of programming and which are not present in our approach. Our perspective stresses the logical (indeed, the "implicative") nature of subtyping and, for now, it presents only the "pure logic". It takes care of the introduction and elimination of universal quantification, which are not present in the other approaches except in that of Mitchell [Mit88]. In a sense, the present paper may be seen as a proof-theoretic analysis of Mitchell's axiomatic approach (see section 7). Further extensions, besides the addition of base types (section 8), may be a reason for further work.

The word "coercion" occurs in varying contexts. For example, in imperative and object-oriented programming languages such as C and C++, coercions are known as "casting functions", whereby variables of one datatype (e.g., boolean) are "converted" or "cast" to another datatype (e.g., integer). In some languages, such conversions may actually change the underlying representation of the variable's contents, in which case the conversion is done at run-time when the contents are known. More semantic interpretations characterise coercions with respect to identity functions, which do not imply a change to underlying representations. Coercions are used in this sense in [Mit88] for example, where they are known as "retyping functions". In practice, the conversions implied by such coercions are performed statically at compile-time, for type-checking, etc.

1.1.2 Models

As regards the key covariance/contravariance issue, the semantic problem should be clear. In a naive way, one may interpret "σ is a subtype of τ" as "σ is a subset of τ" or "σ can be *identically injected* into τ". This set-theoretic understanding is usually expressed by the following rule, known as *subsumption* (Cardelli): if N has type σ and σ is a subtype of τ, then N also has type τ. However, in Set Theory, the rule ($\rightarrow$) is not realized, as there is no way to *inject* $\tau \rightarrow \rho$ into $\sigma \rightarrow \rho$ when σ is a *subset* of τ. There are, of course, several possible injections (by trivially extending functions) of $\sigma \rightarrow \rho$ into $\tau \rightarrow \rho$, but this is the opposite of what is desired. Even Category Theory, where Hom functors are contravariant on the left, doesn't help: if "subtype" is interpreted as "subobject", there is no way to extend a *monomorphism* $m : \sigma \rightarrow \tau$ to a *monomorphism* from $C[\tau, \rho]$ to $C[\sigma, \rho]$ or, in a Cartesian Closed Category, from ρ^τ to ρ^σ.

An early solution was proposed in [BL90] by constructing a specific categorical interpretation with "set-theoretic features": the model of Partial Equivalence Relations (PER) accommodates Cartesian Closure as well as subtyping when $\sigma \leq \tau$ is interpreted as "σ is a subrelation, a subset of pairs, of τ". Indeed, the PER model also provides an interpretation of higher order quantification as closure under indexed products. (Except for a better understanding of this informal introduction, the reader will need no knowledge of PER models nor of Category Theory in the technical parts of this paper.) The overall categorical construction of PER, as a model of higher order lambda-calculus, in particular, Girard's system F [Gir71], works because it is embedded in a constructive approach to Set Theory (the category of ω-Sets [LM91] or the Effective Topos [Hyl82]).

In spite of the categorical relevance of the Effective Topos (and of other categories which interpret higher order lambda-calculi) and the proof-theoretic accounts in [BCGS91, CMMS91, CG92], up to now, there has been no general categorical or purely logical understanding of subtyping (although a recent, and complex, categorical frame has been proposed in [Jac95] and the axiomatic approach in [Mit88] is also a "complete logic" for subtyping). In this paper, we propose a sequent calculus of subtyping, as a fragment of intuitionistic (linear) second order propositional calculus. In particular, we focus on the introduction and elimination rules for $\forall$, which are at the core of second order systems, and on a "cut-elimination" theorem. This is a relevant property for the partial order of subtyping, as the "cut" rule corresponds to transitivity (see also [PS94]).

The idea is that one can give an obvious logical (constructive) understanding of "σ is a subtype of τ" as "σ implies τ", or more precisely, as "σ entails τ" ($\sigma \vdash \tau$). Note first that, with this interpretation, the usual contra(co)-variance rule for $\rightarrow$ makes perfect sense: if σ entails σ' and τ' entails τ, then $\sigma' \rightarrow \tau'$ entails $\sigma \rightarrow \tau$. Moreover, if terms in σ may also be in τ, then this should be possible using some sort of effective transformation: either the identity or a "suitable" coercion, as an effective map from σ to τ. Thus, by the Curry-Howard isomorphism, subtyping is a special case of intuitionistic implication: a computation from σ to τ is a proof of $\sigma \vdash \tau$. But which special case? And how to characterize it? Coercions shouldn't be arbitrary maps. Of course, the simplest idea would be to assume that they are identities. This makes no sense though in typed frameworks: if $\sigma \leq \tau$ but σ and τ are different, there is no identity from σ to τ. This is so both in models and in theories. A way out is suggested by the PER model. PER models are constructed over an underlying model of the type-free lambda calculus. Indeed, any model of partial combinatory logic may suffice, see [Hyl88] say, and in particular Kleene's $(\omega, .)$: in this case, $n.m$ stays for the n-th index or Turing Machine applied to m.

Thus, in the PER model, terms are interpreted as equivalence classes of elements of ω. More precisely, define the erasure of a typed term to be the type-free term obtained by erasing all type information. Then, a term is interpreted by the equivalence class of (the interpretation of) its erasure. This allows second order quantification to be interpreted as intersection, since the intersection is isomorphic to a categorical product (see [LM91]). In short, in the model in [BL90] (more precisely in the variant of it in [CL91], section 4), a coercion c from σ to τ transforms each $\{n\}_\sigma$ into $\{n\}_\tau$, where $\{n\}_\sigma$ denotes the equivalence class of n in σ and $\{n\}_\sigma$ is generally smaller than $\{n\}_\tau$. Thus, c is computed, in particular, by any index i of the identity function; equivalently, c is represented by $\{i\}_{\sigma\rightarrow\tau}$, since $\{i\}_{\sigma\rightarrow\tau}.\{n\}_\sigma = \{i.n\}_\tau = \{n\}_\tau$. Clearly, $\{i\}_{\sigma\rightarrow\tau}$ contains many more elements than the indices of the identity function on ω, when σ or τ are different from ω. Using erasures in order to interpret typed terms, then this suggests that syntactic coercions are typed terms, different, in general, from the identity, but whose *erasure* is equal to the identity $\lambda x.x$. This idea is nicely used in [Mit88]. It will give a syntactic completeness theorem for our "logic of subtyping" described next.

1.2 Subtyping as restricted linear implication

The logical frame we use here is intuitionistic second-order propositional logic. The intended meaning of $\sigma \vdash \tau$ is that σ is contained in τ. An obvious axiom and the contra(co)-variance rule for $\rightarrow$ are the first requests for a logic of subtyping:

(ax) $\sigma \vdash \sigma$ (→) $\dfrac{\sigma' \vdash \sigma \qquad \tau \vdash \tau'}{\sigma \to \tau \vdash \sigma' \to \tau'}$

Consider now a logical interpretation of second order $\forall$. Assume that σ contains X free and that from a specific instance of σ (with ρ substituted for X say), one can deduce τ. Then, from $\forall X.\sigma$ one can, a fortiori, deduce τ. This is the ($\forall$ left) rule of Gentzen's sequent calculus. A semantic understanding of this second order deduction, can be given in the PER model of subtyping: if a specific instance of a family of types is a subtype of τ, then, in the model, the intersection of the entire family is a subtype of τ.

($\forall$ left) $\dfrac{[\rho/X]\sigma \vdash \tau}{\forall X.\sigma \vdash \tau}$

where $[\rho/X]\sigma$ stands for the type resulting from the (capture-avoiding) substitution of type ρ for X in type σ.

Moreover, if σ entails τ, and σ does not contain X free, then σ also entails $\forall X.\tau$. This is Gentzen's ($\forall$ right) rule. Semantically, if σ is a subtype of τ and σ does not depend on X, then σ is also a subtype of the intersection of all τs over X:

($\forall$ right) $\dfrac{\sigma \vdash \tau}{\sigma \vdash \forall X.\tau}$

* *for X not free in σ*

Recall now that the principal idea here is that the embedding of a type into another should be very simple: indeed, as close to the identity as possible in a typed language. Identities are linear maps, to say the least, as our system will be a fragment of the linear sequent calculus. Even more so: we allow only one premise in a sequent $\sigma \vdash \tau$, as even the swapping of inputs is forbidden. Thus, in order to deal with nested implications, we generalize ($\forall$ right) to:

($\forall_{n\geq 0}$ right) $\dfrac{\sigma \vdash \tau_1 \to \ldots(\tau_n \to \tau)\ldots)}{\sigma \vdash \tau_1 \to \ldots(\tau_n \to \forall X.\tau)\ldots)}$

* *for X not free in σ nor in $\tau_1, \ldots, \tau_n$*

($\forall_n$ right) is a family of rules indexed by $n \geq 0$. Note that, if more than one premise was allowed, ($\forall_n$ right) would be the curried variant of ($\forall$ right) with n premises.

These four rules are all we need. The reader may wonder what happened to a fundamental property of subtyping, that is, to transitivity. Indeed, we will *prove* that $\vdash$ is a partial order, thus, in particular that it is transitive and anti-symmetric. But transitivity is just a (cut) rule:

(cut) $\dfrac{\sigma \vdash \tau \qquad \tau \vdash \rho}{\sigma \vdash \rho}$

Proving transitivity will thus be the proof of admissibility for the rule (cut), that is, that each time the premises are derivable, then the consequence is derivable too. Or, equivalently, that the system extended with (cut) has the cut-elimination property. Notice now that the proof one can "eliminate cuts" is non trivial for weak systems (as we will discuss at length below), as, in general, these results and proofs are not inherited to subsystems.

Observe that $\forall$ is introduced to the left and to the right of entailment by two separate rules while $\rightarrow$ is symmetrically introduced both to the left and to the right of entailment by a single rule, the familiar $(\rightarrow)$ rule defined previously.

Cut-elimination is a fundamental property in constructive logical systems. It guarantees consistency as it shows that each derivation can be given a "minimal" structure. In the various lambda-calculi, it relates deductions to computations as cut-elimination corresponds to β-reduction. In those systems, (cut) is a non-primitive rule, as it corresponds to the sequential application of ($\rightarrow$-introduction) and ($\rightarrow$-elimination) rules; moreover, it is usually considered as a (side) consequence of normalization. In contrast to this, for the purposes of subtyping, (cut) is as basic as transitivity.

The seemingly simple logical system above will be shown to be complete and coherent as a logic for deriving subtyping relations. By completeness, we mean that $\sigma \vdash \tau$ is derivable iff there is a term of type $\sigma \rightarrow \tau$ that erases to the identity. We define a coercion to be such a term. By a result in [Mit88] this will also guarantee completeness with respect to subtyping in all PER models, in the sense of [BL90].

Coherence will mean that derivability of $\sigma \vdash \tau$ implies the existence of a *unique* coercion from σ to τ. One of its consequences will be anti-symmetry. Coherence will be easily shown in the simple four rule system, while it requires *cut-elimination* if proved in the system extended with (cut).

Once the formal system is fully written down, with proof-terms displayed (see section 3), the next thing to be described is term equality. The notion of equality we use here may be viewed the "generalized dual" of an early result of Girard's [Gir71]: in system F, there is no definable term that discriminates between types. Namely, there is no definable term J_σ such that J_σ applied to type ρ is 1 if $\sigma = \rho$, and is 0 if $\sigma \neq \rho$. This idea was taken up in [LMS93] by extending system F with the following axiom:

$$\text{(Axiom C)}\quad \frac{M : \forall X.\sigma}{M\tau_1 = M\tau_2}$$

**for X not free in* σ

Intuitively, as there are no type discriminators, (Axiom C) forces terms of universally quantified type, whose outputs live in the same type, to be constant. System F extended with (Axiom C) satisfies the Genericity Theorem (see [LMS93]), which states that if two second-order functions (of the same type) coincide on an input type, then they are, in fact, the same function. Equivalently, types are generic inputs to second-order functions.

(Axiom C) was proposed following [Gir71] and independently of [CMMS91], where the system $F_{\leq}$ is defined which includes the following inference rule. This rule is more general than (Axiom C) by a crucial use of subsumption:

$$\text{(eq appl2)}\quad \frac{M : \forall X.\sigma \qquad [\tau_1/X]\sigma \ \leq\ \mu \qquad [\tau_2/X]\sigma \ \leq\ \mu}{M\tau_1 = M\tau_2 : \mu}$$

Observe that M gives outputs in (possibly) different types $[\tau_1/X]\sigma$ and $[\tau_2/X]\sigma$. Then, intuitively, (eq appl2) says that if these two types (of outputs) have a common supertype μ, then the outputs are equal *when seen as elements of* μ. Thus, in particular, if σ does not contain X free, one obtains (Axiom C).

Note that (Axiom C) is valid in all proper models of system F, in particular in all "parametric models" in the sense of Reynolds ([MR92, ACC93]). Moreover, (eq appl2) holds in the only semantic models of system F with subtyping, namely in PER models, of course with the intended coercions. The relevance of (eq appl2) is that it allows one to prove the (categorical) *universality* of key definable constructs in System F (binary products, coproducts, existentials, etc.).

However, (eq appl2) relies implicitly on the subsumption rule, i.e., if $M : \sigma$ and $\sigma \leq \tau$, then $M : \tau$. And, as already pointed out, subsumption has neither type-theoretic nor categorical meaning, even though it may have solid practical motivations and intuitive meaning. Subsumption may be avoided in (eq appl2) if coercions are used explicitly as follows:

$$\frac{M : \forall X.\sigma \qquad y_1 : [\tau_1/X]\sigma \vdash N_1 : \mu \qquad y_2 : [\tau_2/X]\sigma \vdash N_2 : \mu}{[M\tau_1/y_1]N_1 = [M\tau_2/y_2]N_2}$$

This "coercion version" of (eq appl2) will be used in our type-theory as an interplay between subtyping and equality.

In conclusion then, our logic of subtyping will be based on the simple four rule sequent calculus presented previously, and the proof terms will satisfy the usual equational rules plus a coercion version of (eq appl2).

2 System F

We first recall System F [Gir71]. The language has two kinds of expression, *types* and *terms*, defined by the following syntax:

$$\begin{array}{lrcl} \text{(Types)} & \sigma & ::= & X \mid \sigma \to \tau \mid \forall X.\sigma \\ \text{(Terms)} & M & ::= & x \mid \lambda x{:}\sigma.M \mid MN \mid \lambda X.M \mid M\tau \end{array}$$

We will use:

σ, τ, ρ, μ for types	M, N, P, Q, R for terms
X, Y, Z for type variables	x, y, z for term variables

An *environment* Γ is a set of term variables with their types. We write $\Gamma, x\!:\!\sigma$ to extend Γ with a new term variable x of type σ, where x must not already occur in Γ.

We use the notation $\Gamma \vdash_F M : \sigma$ for type assignment in system F. The following rules define valid type assignments.

System F

$$\text{(ax)} \quad \Gamma, x\!:\!\sigma \vdash_F x : \sigma$$

$$(\to \text{intro}) \quad \frac{\Gamma, x\!:\!\sigma \vdash_F M : \tau}{\Gamma \vdash_F \lambda x\!:\!\sigma.M : \sigma \to \tau} \qquad (\to \text{elim}) \quad \frac{\Gamma \vdash_F M : \sigma \to \tau \qquad \Gamma \vdash_F N : \sigma}{\Gamma \vdash_F MN : \tau}$$

$$(\forall \text{ intro}) \quad \frac{\Gamma \vdash_F M : \sigma}{\Gamma \vdash_F \lambda X.M : \forall X.\sigma} \qquad (\forall \text{ elim}) \quad \frac{\Gamma \vdash_F M : \forall X.\sigma}{\Gamma \vdash_F M\tau : [\tau/X]\sigma}$$

* *for X not free in the type of any free term variable in M*

We write $FV(M)$ for the set of free type and term variables in M, and $[N/x]$ and $[\sigma/X]$ in prefix position for term and type substitution, respectively, with usual renaming of bound variables to avoid capture. Reduction of terms is defined as usual by the closure of the following rules:

$$(\beta_1) \quad (\lambda x\!:\!\sigma.M)N \longrightarrow_{\beta 1} [N/x]M \qquad (\beta_2) \quad (\lambda X.M)\tau \longrightarrow_{\beta 2} [\tau/X]M$$

$$(\eta_1) \quad \lambda x\!:\!\sigma.Mx \longrightarrow_{\eta 1} M \qquad (\eta_2) \quad \lambda X.MX \longrightarrow_{\eta 2} M$$

* *for $x \notin FV(M)$* * *for $X \notin FV(M)$*

We will write $\longrightarrow_{\beta\eta}$ for the transitive closure of all four reductions, $\longrightarrow_{\beta}$ for the closure of just β_1 and β_2; and $\longrightarrow_{\eta}$ for the closure of η_1 and η_2. We will also write "$\beta\eta$-nf" for normal form with respect to all four reduction relations, and "β-nf" or "nf$_\beta$" for normal form with respect to β_1 and β_2.

Different equality relations on terms are defined by the compatible, reflexive, transitive closure of the different reduction relations. In particular, we will write $M =_{\beta\eta} N$ for equality with respect to $\longrightarrow_{\beta\eta}$, and $M =_{\eta} N$ for equality with respect to $\longrightarrow_{\eta}$. We reserve the notation $M \equiv N$ for syntactic identity of terms up to α-conversion, and for types, equality, written $\sigma = \tau$, is just syntactic identity up to α-conversion.

3 System $Co^{\vdash}$

3.1 System $Co^{\vdash}$

We now define our sequent calculus of subtyping, referred to as System $Co^{\vdash}$ or just $Co^{\vdash}$ (pronounced "co" for coercions). We will use $\vdash_{co}$ for entailment in this system.

We give two equivalent presentations of $Co^{\vdash}$. The first presentation gives only the types involved in each judgment, which are of the form $\sigma \vdash_{co} \tau$. This presentation emphasizes the intended subtyping relation between types ($\sigma \vdash_{co} \tau$ can be read "σ is a subtype of τ") but, of course, the calculus may be considered independently of this interpretation by referring just to its logical significance.

System $Co^{\vdash}$ (unlabeled)

(ax) $\quad \sigma \vdash_{co} \sigma$

($\rightarrow$) $\quad \dfrac{\sigma' \vdash_{co} \sigma \qquad \tau \vdash_{co} \tau'}{\sigma \rightarrow \tau \vdash_{co} \sigma' \rightarrow \tau'}$

($\forall$ left) $\quad \dfrac{[\rho/X]\sigma \vdash_{co} \tau}{\forall X.\sigma \vdash_{co} \tau}$

($\forall_{0 \leq n}$ right) $\quad \dfrac{\sigma \vdash_{co} \tau_1 \rightarrow \ldots(\tau_n \rightarrow \tau)\ldots)}{\sigma \vdash_{co} \tau_1 \rightarrow \ldots(\tau_n \rightarrow \forall X.\tau)\ldots)}$

* *for X not free in σ nor in $\tau_1, \ldots, \tau_n$*

In the second presentation of the system, we label each type with a term, yielding judgments of the form $x : \sigma \vdash_{co} M : \tau$. We will refer to the first presentation as the "unlabeled" system, to the second as the "labeled" system.

System $Co^{\vdash}$ (labeled)

(ax) $\quad x:\sigma \vdash_{co} x:\sigma$

(→)
$$\frac{x':\sigma' \vdash_{co} M:\sigma \qquad y:\tau \vdash_{co} N:\tau'}{x:\sigma\to\tau \vdash_{co} \lambda x':\sigma'.[xM/y]N \;:\; \sigma'\to\tau'}$$

(∀ left)
$$\frac{y:[\rho/X]\sigma \vdash_{co} M:\tau}{x:\forall X.\sigma \vdash_{co} [x\rho/y]M:\tau}$$

($\forall_{0\le k\le n}$ right)
$$\frac{x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k.M \;:\; \tau_1\to\ldots(\tau_n\to\tau)\ldots)}{x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k\ldots\lambda x_n:\tau_n.\lambda X.M x_{k+1}\ldots x_n \;:\; \tau_1\to\ldots(\tau_n\to\forall X.\tau)\ldots)}$$

* *for X not free in σ*
nor in $\tau_1,\ldots,\tau_n$,
for M not of the form $\lambda y.M'$,
for $x_{k+1},\ldots,x_n$ fresh

($\forall_{0\le n<k}$ right)
$$\frac{x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k.M \;:\; \tau_1\to\ldots(\tau_n\to\tau)\ldots)}{x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_n:\tau_n.\lambda X.\lambda x_{n+1}:\tau_{n+1}\ldots\lambda x_k:\tau_k.M \;:\; \tau_1\to\ldots(\tau_n\to\forall X.\tau)\ldots)}$$

* *for X not free in σ*
nor in $\tau_1,\ldots,\tau_n$,
for M not of the form $\lambda y.M'$
for $\tau \equiv \tau_{n+1}\to\ldots(\tau_k\to\tau')\ldots)$

We will shortly see that the two cases of ($\forall_n$ right) above are disjoint.[1] In the rest of the paper, we will refer to judgments of either presentation (labeled and unlabeled) of $Co^{\vdash}$ as "sequents". We will use S for sequents and α,γ for derivations of sequents.

Many of the proofs in this paper are by induction on the *size* of a derivation. This notion of size is defined as the total number of applications of rules in a derivation.

In this work, a coercion is defined as follows:

Definition (Coercion) *A sequent $x:\sigma \vdash_{co} M:\tau$ is a coercion from σ to τ iff it is derivable in $Co^{\vdash}$.*

As a linguistic shorthand, we will also refer to the term M as a coercion when the sequent $x:\sigma \vdash_{co} M:\tau$ is derivable.

To illustrate $Co^{\vdash}$, observe that $\perp = \forall X.X$ (the empty type) is provably a subtype of all types. The corresponding coercion is obtained from the following derivation:

$$\frac{y:[\sigma/X]X \vdash_{co} y:\sigma}{x:\forall X.X \vdash_{co} x\,\sigma:\sigma}\,(\forall\text{ left})$$

[1]Tiuryn, in [Tiu96], presents ($\forall_n$ right) in another, slightly simpler form.

Lemma 1 (Structure of coercions) *Assume that* $x:\sigma \vdash_{co} M:\tau$. *Then, M has exactly one free term variable which is x, and x occurs exactly once in M and always at head position. Furthermore, each bound term variable occurs exactly once in the body of M.*

Proof: By induction on the size of the derivation of $x:\sigma \vdash_{co} M:\tau$. ∎

This lemma shows that, in particular, coercions are linear terms (in the usual sense).

Lemma 2 (Coercions are System F functions) *If* $x:\sigma \vdash_{co} M:\tau$ *then* $x:\sigma \vdash_{F} M:\tau$.

Proof: By induction on the size of the derivation of $x:\sigma \vdash_{co} M:\tau$. ∎

Lemma 3 (Coercions are in β-nf) *If* $x:\sigma \vdash_{co} M:\tau$ *then M is in β-nf.*

Proof: By induction on the size of the derivation of $x:\sigma \vdash_{co} M:\tau$, as no inference rule creates a β-redex. ∎

This lemma shows that the terms labeling the two cases of ($\forall_n$ right) are unambiguously defined since M, in the assumption, must be in β-nf. Note too that, although ($\rightarrow$) and ($\forall$ left) may introduce η-redexes, the system is easily shown to be closed under η-reduction. It is also closed under those η-expansions that do not introduce β-redexes.

Lemma 4 *For M in β-nf, assume that* $x:\sigma \vdash_{F} M:\tau$ *and* $M \longrightarrow_{\eta} M'$. *Then,* $x:\sigma \vdash_{co} M:\tau$ *iff* $x:\sigma \vdash_{co} M':\tau$.

Proof: From left to right (closure under η-reduction), by analysis of the derivation of $x:\sigma \vdash_{co} M:\tau$. From right to left (closure under η-expansions that do not introduce β-redexes), by induction on the structure of M'. ∎

Clearly, $Co^{\vdash}$ is strictly weaker than System F. Consider, for example:

$$x:\forall X.X \vdash_{F} x(\forall X.X \rightarrow \forall X.X)x : \forall X.X$$

but

$$x:\forall X.X \not\vdash_{co} x(\forall X.X \rightarrow \forall X.X)x : \forall X.X$$

Even restricting terms to linear ones, and environments to those containing exactly one variable, is not enough to produce a coercion. For example,

$$x:\sigma \rightarrow (\tau \rightarrow \rho) \vdash_{F} \lambda y:\tau.\lambda z:\sigma.(x\ z\ y) : \tau \rightarrow (\sigma \rightarrow \rho)$$

is not a coercion. Theorem 14 (completeness) will characterize those System F functions that are coercions.

By the Curry-Howard isomorphism, $Co^{\vdash}$ is thus a proper subsystem of intuitionistic (linear) second-order propositional logic. This is only natural since coercions are intended to represent "inclusions" of types. Clearly, there is no reason why arbitrary F-entailment (or even isomorphisms of types) should be interpreted as inclusion.

3.2 Transitivity of $\vdash_{co}$

In section 6, we will show that $\vdash_{co}$ is a transitive relation. That is, in the unlabeled system, the rule:

$$\text{(cut)} \quad \frac{\sigma \vdash_{co} \tau \qquad \tau \vdash_{co} \rho}{\sigma \vdash_{co} \rho}$$

is admissible.

Its labeled version needs some discussion. A "naive" variant would be:

$$\frac{x\!:\!\sigma \vdash_{co} M : \tau \qquad y\!:\!\tau \vdash_{co} N : \rho}{x\!:\!\sigma \vdash_{co} [M/y]N : \rho}$$

But note that the term $[M/y]N$ may not be a coercion; in particular, it may not be in β-nf (cf. Lemma 3). However, *without* making use of the strong normalization property of System F, we can prove that $[M/y]N$ normalizes:

Lemma 5 *Assume that $x\!:\!\sigma \vdash_{co} M : \tau$ and $y\!:\!\tau \vdash_{co} N : \rho$. Then, there exists a unique, finite path of β-reductions from $[M/y]N$. Consequently, there exists a unique β-nf of $[M/y]N$.*

Proof: By the linearity of coercions (Lemma 1), M and N are linear, and so the term $[M/y]N$ is also linear (remember that the only free term variable of M is x and of N is y). Observe then that, because M and N are in β-nf (Lemma 3), only one β-redex may exist in $[M/y]N$. And each subsequent β-reduction from $[M/y]N$ preserves linearity and may introduce only one new β-redex. Furthermore, each such β-reduction decreases the number of λs, say n, originally in $[M/y]N$. Thus, there is a unique path, with at most n steps, of β-reductions from $[M/y]N$. ■

The existence and unicity of the β-nf of $[M/y]N$ now motivates the following alternative version of the labeled (cut)-rule, where $\text{nf}_\beta([M/y]N)$ means the β-nf of $[M/y]N$:

$$\text{(cut)} \qquad \frac{x\!:\!\sigma \vdash_{co} M : \tau \qquad y\!:\!\tau \vdash_{co} N : \rho}{x\!:\!\sigma \vdash_{co} \text{nf}_\beta([M/y]N) : \rho}$$

Our aim will be to show that $x\!:\!\sigma \vdash_{co} \text{nf}_\beta([M/y]N) : \rho$ is actually a coercion, that is, derivable in $Co^{\vdash}$; or, in other words, that the rule above is admissible.

Note that (cut) is a "cut" rule in the usual sense of sequent calculi. Thus, proving the admissibility of (cut) is equivalent to proving a *cut-elimination* theorem for the extended system $Co^{\vdash}$ plus (cut). Observe too that subject reduction holds trivially in the extended system: all terms in $Co^{\vdash}$ plus (cut) are in β-nf.

3.3 Equality $=_{\eta co}$

Equality of coercions is defined, essentially, by η-equality plus a coercion version of the $F_{\leq}$ [CMMS91] rule (eq appl2). We write $x : \sigma \vdash_{co} M =_{\eta co} N : \tau$ to mean that M and N are equal coercions from σ to τ. This relation is defined as follows.

Definition (Equality of coercions) *$=_{\eta co}$ is the least equivalence relation generated by α-convertibility and the two rules*

$$\text{(eq }\eta\text{)} \qquad \frac{x : \sigma \vdash_{co} M : \tau \qquad x : \sigma \vdash_{co} M' : \tau \qquad M =_{\eta} M'}{x : \sigma \vdash_{co} M =_{\eta co} M' : \tau}$$

$$\text{(eq appl2 co)} \qquad \frac{y_1 : [\tau_1/X]\sigma \vdash_{co} N_1 : \mu \qquad y_2 : [\tau_2/X]\sigma \vdash_{co} N_2 : \mu}{x : \forall X.\sigma \vdash_{co} [x\tau_1/y_1]N_1 \ =_{\eta co} \ [x\tau_2/y_2]N_2 \ : \ \mu}$$

plus two other rules, (eq $\rightarrow$) and (eq $\forall_n$ right), which state, respectively, that ($\rightarrow$) and ($\forall_n$ right) preserve equality of coercions. (eq $\rightarrow$) and (eq $\forall_n$ right) are given in full in the appendix.

Note that (eq appl2 co) implies that ($\forall$ left) preserves equality of coercions: given $y : [\rho/X]\sigma \vdash_{co} M =_{\eta co} N : \tau$, apply (eq appl2 co) with $\tau_1 \equiv \tau_2 \equiv \rho$ and $N_1 \equiv M$ and $N_2 \equiv N$ to obtain $x : \forall X.\sigma \vdash_{co} [x\rho/y]M \ =_{\eta co} \ [x\rho/y]N \ : \ \tau$.

The rules (eq $\rightarrow$) and (eq $\forall_n$ right) are derivable in system F using β-convertibility. Rule (eq appl2 co), on the other hand, is not derivable in system F as it equates terms that are not β-convertible, as shown by the following instance of the rule.

Example:

$$\frac{y_1 : [\tau_1/X]Y \ \vdash_{co} \ y_1 : Y \qquad y_2 : [\tau_2/X]Y \ \vdash_{co} \ y_2 : Y}{x : \forall X.Y \ \vdash_{co} \ [x\tau_1/y_1]y_1 \ =_{\eta co} \ [x\tau_2/y_2]y_2 \ : \ Y} \text{(eq appl2 co)}$$

Thus, $x\tau_1 =_{\eta co} x\tau_2$ whereas $x\tau_1 \neq_{\beta\eta} x\tau_2$ in general, i.e., they are not equal in System F. Clearly, though, $x\tau_1$ and $x\tau_2$ are equal in $F_{\leq}$ [CMMS91] using the corresponding "non-coercion" $F_{\leq}$ rule (eq appl2); see section 1.2.

Here are two examples to illustrate $=_{\eta co}$ (recall that $\perp = \forall X.X$). Note that these equalities are not provable in system F.

Example 1: $x : \perp \ \vdash_{co} \ x =_{\eta co} x\perp \ : \ \perp$

$$\cfrac{\cfrac{\cfrac{y_1 : [X/X]X \ \vdash_{co} \ y_1 : X \qquad \cfrac{y : [X/X]X \ \vdash_{co} \ y : X}{y_2 : [\perp/X]X \ \vdash_{co} \ y_2 X : X}\text{($\forall$ left)}}{x : \perp \ \vdash_{co} \ xX \ =_{\eta co} \ x\perp X \ : \ X}\text{(eq appl2 co)}}{x : \perp \ \vdash_{co} \ \lambda X.xX \ =_{\eta co} \ \lambda X.x\perp X \ : \ \perp}\text{(eq $\forall_0$ right)}}{x : \perp \ \vdash_{co} \ x \ =_{\eta co} \ x\perp \ : \ \perp}\text{(eq η)}$$

Example 2: $x:\forall X\,.\,X\to X \ \vdash_{co}\ \lambda y:\bot\,.\,\lambda X\,.\,xX(yX)\ =_{\eta co}\ x\bot\ :\ \bot\to\bot$

$$\dfrac{\dfrac{\dfrac{y':X \vdash_{co} y':X}{y:\bot \vdash_{co} yX : X}\,(\forall\text{ left}) \qquad x':X \vdash_{co} x':X}{y_1:X\to X \vdash_{co} \lambda y:\bot\,.\,y_1(yX) : \bot\to X}\,(\to) \qquad \dfrac{y:\bot \vdash_{co} y:\bot \quad \dfrac{y':X \vdash_{co} y':X}{x':\bot \vdash_{co} x'X : X}\,(\forall\text{ left})}{y_2:\bot\to\bot \vdash_{co} \lambda y:\bot\,.\,y_2yX : \bot\to X}\,(\to)}{\dfrac{x:\forall X.X\to X \vdash_{co} \lambda y:\bot\,.\,xX(yX)\ =_{\eta co}\ \lambda y:\bot\,.\,x\bot yX\ :\ \bot\to X}{x:\forall X.X\to X \vdash_{co} \lambda y:\bot\,.\,\lambda X\,.\,xX(yX)\ =_{\eta co}\ \lambda y:\bot\,.\,\lambda X\,.\,x\bot yX\ :\ \bot\to\bot}\,(\text{eq }\forall_1\text{ right})}\,(\text{eq appl2 co})$$

As well as equality of terms, we will need equality of derivations, defined by the following three relations:

Definition (Equality of derivations) *Let α_1 be a derivation of $x:\sigma \vdash_{co} M:\tau$ and α_2 a derivation of $x:\sigma \vdash_{co} N:\tau$. We write:*
- $\alpha_1 = \alpha_2$ *iff* $M \equiv N$
- $\alpha_1 =_\eta \alpha_2$ *iff* $x:\sigma \vdash_{co} M =_\eta N$
- $\alpha_1 =_{\eta co} \alpha_2$ *iff* $x:\sigma \vdash_{co} M =_{\eta co} N$.

3.4 Semantics of $=_{\eta co}$

Clearly, the semantics of rules (eq $\to$) and (eq $\forall_n$ right) pose no problems. However, the semantics, indeed the consistency, of (eq appl2 co) deserves closer attention. The following remarks are intended for the reader familiar with PER models.

Rule (eq appl2 co) is valid in the PER model of subtyping [BL90]. More precisely, it is valid in the interpretation of subtyping with explicit coercions, given in [CL91] for the system $Quest_C$. This can be seen as follows. Recall that, under an environment ξ, the interpretation of $M:\rho$ is given by the equivalence class $\{[\![erase(M)_\xi]\!]\}_\rho$, where, by an abuse of language, we use the same name for both a type ρ and its meaning as a p.e.r.. (See the introduction and [CL91] on how type-free and typed environments are related; in short, a type-free environment ξ "picks out" an element of the equivalence class of the typed ξ'). Assume now that $y_i : [\tau_i/X]\sigma$ for $i=1,2$, and $x:\forall X.\sigma$. If both $[\tau_i/X]\sigma$ for $i=1,2$ are subtypes of μ, then the equivalence classes $\{\xi(y_i)\}_{[\tau_i/X]\sigma}$ are coerced to the (larger) equivalence classes $\{\xi(y_i)\}_\mu$ for $i=1,2$. Note now that, for $x:\forall X.\sigma$, one has $x\tau_i : [\tau_i/X]\sigma$ and $erase(x\tau_i) = erase(x) = x$. Then, by the interpretation of polymorphic application (see [LM91]), $\{\xi(x)\}_{\forall X.\sigma}\,.\,\tau_i = \{\xi(x)\}_{[\tau_i/X]\sigma}$, which is also equal to the interpretation of $x\tau_i : [\tau_i/X]\sigma$.

The validity of the premises of (eq appl2 co) means that, in the model, N_i coerces the meaning of any term in $[\tau_i/X]\sigma$ into an equivalence class of μ. In particular, both interpretations $\{\xi(x)\}_{[\tau_i/X]\sigma}$ of $x\tau_i : [\tau_i/X]\sigma$, $i=1,2$, are coerced by N_i to the *same* equivalence class $\{\xi(x)\}_\mu$, which does not depend on i. This is exactly the validity of the consequence of (eq appl2 co).

As recalled in the introduction, the coercions N_1, N_2 are interpreted by functions computed (also) by indexes of the identity function. In general, though, they are not themselves identities and $\{\xi(y_i)\}_{[\tau_i/X]\sigma}$ may be strictly smaller than $\{\xi(y_i)\}_\mu$ (and each of the $\{\xi(x)\}_{[\tau_i/X]\sigma}$ may be strictly smaller than $\{\xi(x)\}_\mu$). In [CMMS91], it is said that (eq appl2), which contains no coercions, is valid in PER models. This is correct but only modulo "forgetting" the coercions in the model (as was suggested in [BL90]). However, this does not correspond exactly to the structure of PER models, where coercions are non-identical maps (see [CL91] for a more detailed discussion). Thus, (eq appl2 co) is a more precise formalisation of "truth", as given in PER models, than (eq appl2).

3.5 Pure variable derivations

Definition (Pure variable derivation) *A pure variable derivation is a derivation in which no type variable occurs both free and bound in the same sequent.*

This notion of a pure variable derivation is comparable with that used by Kleene (see [Kle67]).

Lemma 6 *Every derivation in $Co^{\vdash}$ is $=$ to any derivation obtained by safely renaming bound type variables in types and terms, without capturing free type variables.*

Proof: By induction on the size of the derivation. In the case of ($\forall$ left), use the identity $[\rho/X]\sigma = [\rho/X']([X'/X]\sigma)$ where X' does not occur free in σ (otherwise X' would be captured), and in the case of ($\forall_n$ right), uniformly substitute $[X'/X]$ in the derivation of the premise and then use ($\forall_n$ right) with X' instead of X (the side-condition is used). For ($\to$), it is enough to use the induction hypothesis. ∎

Lemma 7 *Every derivation in $Co^{\vdash}$ is $=$ to a pure variable derivation.*

Proof: By induction, using the previous lemma (always choosing fresh bound variables). ∎

To illustrate the utility of pure variable derivations, consider the following derivation:

$$\dfrac{\dfrac{\alpha_l}{x':\sigma' \vdash_{co} M:\sigma} \qquad \dfrac{\dfrac{\alpha_r}{y:\tau \vdash_{co} N:\tau'}}{y:\tau \vdash_{co} \lambda X.N:\forall X.\tau'}\,(\forall_0 \text{ right})}{x:\sigma\to\tau \vdash_{co} \lambda x':\sigma'.\,\lambda X.\,[xM/y]N : \sigma'\to\forall X.\tau'}\,(\to)$$

By the side-condition on ($\forall_0$ right), we know that X is not free in τ. Our aim now is to permute the applications of ($\forall_0$ right) and ($\to$) as follows:

$$\dfrac{\dfrac{\dfrac{\alpha_l}{x':\sigma' \vdash_{co} M:\sigma} \qquad \dfrac{\alpha_r}{y:\tau \vdash_{co} N:\tau'}}{x:\sigma\to\tau \vdash_{co} \lambda x':\sigma'.\,[xM/y]N : \sigma'\to\tau'}\,(\to)}{x:\sigma\to\tau \vdash_{co} \lambda x':\sigma'.\,\lambda X.\,[xM/y]N : \sigma'\to\forall X.\tau'}\,(\forall_1 \text{ right})$$

However, this second derivation is only possible if the variable X is not free in σ nor in σ', as required by ($\forall_1$ right). This is the case if the first derivation is a pure variable one, for then the X bound in $\forall X.\tau'$ would be fresh.

From now on, we will work exclusively with pure variable derivations, referring to them as just "derivations".

4 Coherence of $Co^{\vdash}$

By coherence, we mean that a coercion from type σ to type τ is independent of its derivation in $Co^{\vdash}$, in the sense that if a coercion from σ to τ exists, then it is unique, up to $=_{\eta co}$.

In order to prove coherence of $Co^{\vdash}$, we establish two technical results about equal derivations. The first result allows us to permute certain rules in a derivation yielding an equal derivation, and the second introduces the notion of an "atomic" derivation and shows that, for each general $Co^{\vdash}$ derivation, there is an equal atomic $Co^{\vdash}$ derivation.

Lemma 8 *The following pairs of derivations are equal in $Co^{\vdash}$, up to $=$:*

1.
$$\dfrac{\dfrac{\alpha}{S_1}\,(\forall_n\ \mathit{right})}{S_2}\,(\forall\ \mathit{left}) \quad = \quad \dfrac{\dfrac{\alpha}{S_1'}\,(\forall\ \mathit{left})}{S_2}\,(\forall_n\ \mathit{right})$$

2.
$$\dfrac{\alpha_l \qquad \dfrac{\alpha_r}{S_1}\,(\forall_n\ \mathit{right})}{S_2}\,(\to) \quad = \quad \dfrac{\dfrac{\alpha_l \qquad \alpha_r}{S_1'}\,(\to)}{S_2}\,(\forall_{n+1}\ \mathit{right})$$

Proof: By simple consideration of the types and terms involved in each sequent. Case 1 applies to any kind of derivation whereas case 2 requires a pure variable derivation. Indeed, the proof of case 2, permuting ($\forall_n$ right) for $n = 0$ and ($\to$), is given by the examples of pure variable derivations above. Remember that, by the definition of $=$ on derivations, the terms labeling the end sequents of both the left and right derivations above, are identical. ∎

Note that ($\forall_n$ right) becomes ($\forall_{n+1}$ right) when it is permuted downwards with ($\to$). Note also that case 1 applies only when ($\forall_n$ right) is permuted downwards with ($\forall$ left), not conversely.

Definition (Atomic derivation) *An atomic $Co^{\vdash}$ derivation is one in which all axioms are of the form $x : X \vdash_{co} x : X$.*

Lemma 9 *Every derivation is $=_\eta$ to an atomic derivation.*

Proof: Let α be a derivation of $x:\sigma \vdash_{co} M:\tau$. If α is not atomic then an axiom of the form $x:\rho \vdash_{co} x:\rho$ where ρ is not a type variable, is used at a leaf. It suffices to prove the thesis for such axioms. Proceed by case analysis of the structure of ρ.

Case: $\rho = \forall X.\rho'$. Construct then the following atomic derivation:

$$\dfrac{\dfrac{y:[X/X]\rho' \vdash_{co} y:\rho'}{x:\forall X.\rho' \vdash_{co} xX:\rho'}(\forall\text{ left})}{x:\forall X.\rho' \vdash_{co} \lambda X.xX:\forall X.\rho'}(\forall_0\text{ right})$$

where $\lambda X.xX =_\eta x$.

Case: $\rho = \rho_1 \to \rho_2$. Construct the following atomic derivation:

$$\dfrac{y:\rho_1 \vdash_{co} y:\rho_1 \qquad y':\rho_2 \vdash_{co} y':\rho_2}{x:\rho_1 \to \rho_2 \vdash_{co} \lambda y:\rho_1.xy:\rho_1 \to \rho_2}(\to)$$

where $\lambda y:\rho_1.xy =_\eta x$. ∎

Atomic derivations simplify matters in the rest of this section by allowing us to work up to $=$ with atomic derivations, instead of up to $=_\eta$ with general derivations.

In the following lemma, we "transform" atomic derivations to a useful form (for later purposes) by permuting rules as appropriate.

Lemma 10 *Let α be an atomic derivation of $x:\sigma \vdash_{co} M:\tau_1 \to (\ldots(\tau_n \to \forall X.\tau)\ldots)$. Then, there exists an atomic derivation $\alpha' = \alpha$, with α and α' of equal size, and where ($\forall_n$ right) is the last rule used in α'.*

Proof: Proceed by induction on the derivation of α. Clearly, (ax) cannot have been used, and if ($\forall_n$ right) was used last in α, then we are done. In the other two cases, use Lemma 8 to permute rules as follows:

Case: ($\forall$ left) used last. Apply the induction hypothesis to the premise then permute ($\forall$ left) with ($\forall_n$ right) (Lemma 8, case 1).

Case: ($\to$) used last. Apply the induction hypothesis to the left premise then permute ($\to$) with ($\forall_{n-1}$ right), the latter becoming ($\forall_n$ right) in the final derivation (Lemma 8, case 2). ∎

Lemma 11 *Let α_1 and α_2 be two atomic derivations of $x:\sigma \vdash_{co} M:\tau$ and $x:\sigma \vdash_{co} N:\tau$ respectively. Then, there exist atomic derivations $\alpha_1' = \alpha_1$ and $\alpha_2' = \alpha_2$ such that α_1' and α_2' end with the same rule.*

Proof: If α_1 and α_2 end with the same rule, then, trivially, we are done. Otherwise, based on the structure of σ and τ, it is sufficient to consider two cases: either α_1 ends with ($\forall_n$ right) and α_2 with ($\to$), or α_1 ends with ($\forall_n$ right) and α_2 with ($\forall$ left).
In the first case, the end sequent in both derivations must be of the form $x:\sigma \vdash_{co} M:\tau_1 \to \forall X.\tau_2$. Apply then Lemma 10 to α_2 yielding a derivation $\alpha_2' = \alpha_2$ ending with ($\forall_n$ right), and take $\alpha_1' \equiv \alpha_1$. The second case is treated likewise. If α_1 ends

with ($\forall_n$ right), apply Lemma 10 to α_2 yielding a derivation $\alpha_2' = \alpha_2$ ending with ($\forall_n$ right). Take $\alpha_1' \equiv \alpha_1$. ∎

We are now in a position to prove the coherence of $Co^{\vdash}$ derivations. Note that this is where $=_{\eta co}$, and thus the rule (eq appl2 co), are used.

Theorem 12 (Coherence of $Co^{\vdash}$ derivations) *Let α_1 and α_2 be two derivations of $x:\sigma \vdash_{co} M:\tau$ and $x:\sigma \vdash_{co} N:\tau$ respectively. Then, $\alpha_1 =_{\eta co} \alpha_2$.*

Proof: By Lemmas 9 and 11, there exist atomic derivations $\alpha_1' =_\eta \alpha_1$ and $\alpha_2' =_\eta \alpha_2$ such that α_1' and α_2' both end with the same rule. The proof is easy now. However, it uses induction in a peculiar way (which stresses the strength of (eq appl2 co)). Consider the obvious syntactic length of a type. Then the induction is on the length of τ, when $x:\sigma \vdash_{co} M:\tau$ is the final sequent.

Case: ($\rightarrow$) applied last.
Use induction and (eq $\rightarrow$).

Case: ($\forall_n$ right) applied last.
Use induction and (eq $\forall_n$ right).

Case: ($\forall$ left) applied last. In this case, the length of τ is not changed, but we do not need the inductive hypothesis, here, in view of (eq appl2 co)

Indeed, the final steps in α_1' and α_2' are respectively:

$$\dfrac{\dfrac{\alpha_l}{y_1:[\tau_1/X]\sigma \vdash_{co} N_1:\tau}}{x:\forall X.\sigma \vdash_{co} [x\tau_1/y_1]N_1:\tau}\,(\forall \text{ left}) \qquad \dfrac{\dfrac{\alpha_r}{y_2:[\tau_2/X]\sigma \vdash_{co} N_2:\tau}}{x:\forall X.\sigma \vdash_{co} [x\tau_2/y_2]N_2:\tau}\,(\forall \text{ left})$$

Now, use (eq appl2 co) on both premises:

$$\dfrac{\dfrac{\alpha_l}{y_1:[\tau_1/X]\sigma \vdash_{co} N_1:\tau} \qquad \dfrac{\alpha_r}{y_2:[\tau_2/X]\sigma \vdash_{co} N_2:\tau}}{x:\forall X.\sigma \vdash_{co} [x\tau_1/y_1]N_1 =_{\eta co} [x\tau_2/y_2]N_2}\,(\text{eq appl2 co})$$

Hence, by definition, $\alpha_1' =_{\eta co} \alpha_2'$. ∎

It will be an easy corollary of coherence and the transitivity of $\vdash_{co}$ (section 6) that $x:\sigma \vdash_{co} M:\tau$ and $x:\tau \vdash_{co} N:\sigma$ implies that σ is isomorphic to τ. In other words, $\vdash_{co}$ is anti-symmetric up to isomorphism.

5 Completeness of $Co^{\vdash}$

The standard notion of erasure, defined as follows, will serve to characterize coercions.

Definition (Erasure) *The erasure of a polymorphic term to a type-free term is defined by:*

$erase(x) \equiv x$

$erase(\lambda x\!:\!\sigma.M) \equiv \lambda x.erase(M)$ $\qquad erase(MN) \equiv erase(M)erase(N)$

$erase(\lambda X.M) \equiv erase(M)$ $\qquad erase(M\tau) \equiv erase(M)$

It is simple to show that the erasure of a coercion η-reduces to the identity (in the type-free λ-calculus).

Lemma 13 (Coercions erase to identity) *If $x\!:\!\sigma \vdash_{co} M\!:\!\tau$ then $erase(M) \longrightarrow_\eta x$.*

Conversely, we will now show that any System F term in β-nf whose erasure η-reduces to a term variable is a coercion. This will give a complete characterization of coercions. Moreover, it will be a key step in the proof of transitivity of $\vdash_{co}$ and in relating our system to that of Mitchell [Mit88] (section 7). The proof proceeds by syntactic analysis of the normal form.

Theorem 14 (Completeness) *Let M be a term in β-nf such that $x:\sigma \vdash_F M:\tau$ and $erase(M) \longrightarrow_\eta x$. Then, $x\!:\!\sigma \vdash_{co} M:\tau$.*

Proof: It is enough to prove the theorem for M in $\beta\eta$-normal form since, by Lemma 4, coercions are closed under η-expansions that do not introduce β-redexes.

Assume then that M is in $\beta\eta$-normal form. The proof proceeds by induction on the structure of M, and by Lemma 1 on the structure of coercions. The first step consists of identifying those subterms of M in $\beta\eta$-nf and η-reducing to a term variable.

First, by the assumption that M is in $\beta\eta$-nf, M must have the following structure:

$$M \;\equiv\; \lambda\vec{Y}_1 . \lambda y_1\!:\!\tau_1 . \;\ldots\; \lambda\vec{Y}_k . \lambda y_k\!:\!\tau_k . \lambda\vec{Y}_{k+1} . x\vec{\rho}_1 M_1 \ldots \vec{\rho}_n M_n \vec{\rho}_{n+1}$$

with each subterm M_i, for $1 \leq i \leq n$, in $\beta\eta$-nf. Furthermore, the assumption $erase(M) \longrightarrow_\eta x$ implies that $k = n$ and $erase(M_i) \longrightarrow_\eta y_i$ for $1 \leq i \leq n$.

Consider now the assumption $x:\sigma \vdash_F M:\tau$. In the body of M, x occurs as $x\vec{\rho}_1 M_1 \ldots \vec{\rho}_n M_n$ so the type σ of x must have the following structure:

$$\sigma \;=\; \forall\vec{X}_1.(\sigma_1 \rightarrow \;\ldots\; \forall\vec{X}_n.(\sigma_n \rightarrow \forall\vec{X}_{n+1}.\sigma')\ldots)$$

where, for $1 \leq i \leq n+1$, the vectors $\vec{X}_i$ and $\vec{\rho}_i$ are equal in length ($|\vec{X}_i| = |\vec{\rho}_i|$). Moreover, the type of each subterm M_i, $1 \leq i \leq n$, must then be given by:

$$x\!:\!\sigma,\; y_1\!:\!\tau_1,\; \ldots,\; y_n\!:\!\tau_n \vdash_F M_i : [\vec{\rho}_1/\vec{X}_1,\; \ldots,\; \vec{\rho}_i/\vec{X}_i]\sigma_i$$

However, the fact that $erase(M_i) \longrightarrow_\eta y_i$ means that y_i is the only free term variable in M_i. So, by strengthening of F environments (cf. [CMMS91, section 2.3]), we obtain:

$$y_i\!:\!\tau_i \vdash_F M_i : [\vec{\rho}_1/\vec{X}_1,\; \ldots,\; \vec{\rho}_i/\vec{X}_i]\sigma_i$$

Thus, we have shown that the subterms M_i, $1 \leq i \leq n$, are in $\beta\eta$-nf with $erase(M_i) \longrightarrow_\eta y_i$ and $y_i : \tau_i \vdash_F M_i : [\vec{\rho}_1/\vec{X}_1, \ldots, \vec{\rho}_i/\vec{X}_i]\sigma_i$. These are therefore the subterms satisfying the same assumptions as M so, by induction:

$$y_i : \tau_i \vdash_{co} M_i : [\vec{\rho}_1/\vec{X}_1, \ldots, \vec{\rho}_i/\vec{X}_i]\sigma_i \tag{1}$$

In other words, each subterm M_i, $1 \leq i \leq n$, is a coercion.

To show that M is a coercion requires two further facts derivable from the assumptions and the structure of M:

$$M \quad \equiv \quad \lambda\vec{Y}_1 \,.\, \lambda y_1 : \tau_1 \,.\, \ \ldots \ \lambda\vec{Y}_n \,.\, \lambda y_n : \tau_n \,.\, \lambda\vec{Y}_{n+1} \,.\, x\vec{\rho}_1 M_1 \ldots \vec{\rho}_n M_n \vec{\rho}_{n+1}$$

First, since M is well-typed in F by the assumption $x : \sigma \vdash_F M : \tau$, then for $1 \leq i \leq n+1$, each $\vec{Y}_i$ is not free in the type of preceding bound term variables $y_1, \ldots, y_{i-1}$ nor in the type of x. In other words,

$$\vec{Y}_i \text{ not free in } \sigma \text{ nor in } \tau_1, \ldots, \tau_{i-1}, \qquad \text{for } 1 \leq i \leq n+1 \tag{2}$$

Second, given the structure of M, then the type τ of M must have the following form:

$$\tau \quad = \quad \forall\vec{Y}_1.(\tau_1 \rightarrow \quad \ldots \quad \forall\vec{Y}_n.(\tau_n \rightarrow \forall\vec{Y}_{n+1}.\tau')\ldots)$$

where τ' is the type of the body of M, i.e., the type of $x\vec{\rho}_1 M_1 \ldots \vec{\rho}_n M_n \vec{\rho}_{n+1}$. However, since σ is the type of x, the body of M has the type:

$$[\vec{\rho}_1/\vec{X}_1, \ \ldots, \ \vec{\rho}_{n+1}/\vec{X}_{n+1}]\sigma'$$

Since a well-typed term has a unique type in F, we thus obtain the following:

$$\tau' \quad = \quad [\vec{\rho}_1/\vec{X}_1, \ \ldots, \ \vec{\rho}_{n+1}/\vec{X}_{n+1}]\sigma' \tag{3}$$

We now construct a $Co^{\vdash}$ derivation showing that $x : \sigma \vdash_{co} M : \tau$. To improve readability of this complex derivation, we will denote by δ_i the substitution

$$[\vec{\rho}_1/\vec{X}_1, \ \ldots, \ \vec{\rho}_i/\vec{X}_i]$$

Thus, (1) becomes $y_i : \tau_i \vdash_{co} M_i : \delta_i\sigma_i$ and (3) becomes $\tau' = \delta_{n+1}\sigma'$.

Basically, the leaves of the derivation are the sequents (1) plus an axiom that uses (3). Rules ($\rightarrow$) and ($\forall$ left) are then applied alternately to introduce σ on the left and the underlying $\rightarrow$ structure of τ on the right. Finally, ($\forall$ right) is used repeatedly to introduce $\forall\vec{Y}_i$ on the right. Note that the non-freeness side-conditions for these applications of ($\forall$ right) are satisfied by (2).

$$\begin{array}{c l}
\begin{array}{cc} & z:\delta_{n+1}\sigma' \vdash_{co} z:\tau' \\ \cline{2-2} (1)\ \ y_n:\tau_n \vdash_{co} M_n:\delta_n\sigma_n & x_{n+1}:\delta_n(\forall\vec{X}_{n+1}.\sigma') \vdash_{co} x_{n+1}\vec{\rho}_{n+1}:\tau' \end{array} & \begin{array}{l}\text{(ax) by (3)}\\ (\forall\text{ left})\end{array} \\ \hline
z_n:\delta_n(\sigma_n \to \forall\vec{X}_{n+1}.\sigma') \vdash_{co} \lambda y_n:\tau_n\,.\,z_n M_n\vec{\rho}_{n+1} : \tau_n \to \tau' & (\to) \\ \hline
\begin{array}{cc} (1)\ \ y_{n-1}:\tau_{n-1} \vdash_{co} M_{n-1}:\delta_{n-1}\sigma_{n-1} & x_n:\delta_{n-1}(\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')) \vdash_{co} \lambda y_n:\tau_n\,.\,x_n\vec{\rho}_n M_n\vec{\rho}_{n+1} : \tau_n \to \tau' \end{array} & (\forall\text{ left}) \\ \hline
\vdots & (\to) \\
\vdots & \vdots \\
\vdots & (\forall\text{ left}) \\ \hline
z_1:\delta_1(\sigma_1 \to \ldots\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')\ldots) \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\,.\,z_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \tau' & (\to) \\ \hline
x:\forall\vec{X}_1.(\sigma_1 \to \ldots\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')\ldots) \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \tau' & (\forall\text{ left}) \\ \hline
x:\sigma \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\lambda\vec{Y}_{n+1}\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \forall\vec{Y}_{n+1}.\tau' & (\forall_{n+1}\text{ right) due to (2)} \\ \hline
\vdots & (\forall_n\text{ right) due to (2)} \\
\vdots & \vdots \\ \hline
x:\sigma \vdash_{co} \lambda\vec{Y}_1\,.\,\lambda y_1:\tau_1\ \ldots\ \lambda\vec{Y}_n\,.\,\lambda y_n:\tau_n\,.\,\lambda\vec{Y}_{n+1}\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \forall\vec{Y}_1.(\tau_1 \to \ldots\forall\vec{Y}_n.(\tau_n \to \forall\vec{Y}_{n+1}.\tau')\ldots) & (\forall_0\text{ right) due to (2)}
\end{array}$$

Thus, $x:\sigma \vdash_{co} M:\tau$ so M is a coercion. ∎

6 Transitivity of $\vdash_{co}$

We now prove that $\vdash_{co}$ is a transitive relation. More formally, we will prove that the rule:

$$\text{(cut)} \qquad \frac{x:\sigma \vdash_{co} M:\tau \qquad y:\tau \vdash_{co} N:\rho}{x:\sigma \vdash_{co} \mathrm{nf}_\beta([M/y]N):\rho}$$

is admissible in $Co^{\vdash}$.

Lemma 15 *Assume that $x:\sigma \vdash_{co} M:\tau$ and $y:\tau \vdash_{co} N:\rho$. Then, $erase(nf_\beta([M/y]N)) =_\eta nf_\beta(erase([M/y]N))$.*

Proof: Note first that, by an argument similar to that used to prove Lemma 5, there exists a unique β-nf of $erase([M/y]N)$, i.e., $\mathrm{nf}_\beta(erase([M/y]N))$ does indeed exist. The only difference is that there will be no β_2-reductions in the path from $erase([M/y]N)$. Next, by Lemma 5 directly, $\mathrm{nf}_\beta([M/y]N)$ exists. Finally, the erasure of $\mathrm{nf}_\beta([M/y]N)$ is clearly η-equivalent to the unique β-nf of $erase([M/y]N)$. ∎

Theorem 16 (Transitivity) *(cut) is an admissible rule in $Co^{\vdash}$.*

Proof: Assume that $x:\sigma \vdash_{co} M:\tau$ and $y:\tau \vdash_{co} N:\rho$. By Lemma 5, $\mathrm{nf}_\beta([M/y]N)$ exists. Since $x:\sigma \vdash_F [M/y]N:\rho$, then, by subject reduction in System F, $x:\sigma \vdash_F \mathrm{nf}_\beta([M/y]N):\rho$. Furthermore, since coercions erase to the identity (Lemma 13), $erase(M) \longrightarrow_\eta x$ and $erase(N) \longrightarrow_\eta y$. Then:

$$\begin{array}{rcll} erase(\mathrm{nf}_\beta([M/y]N)) & =_\eta & \mathrm{nf}_\beta(erase([M/y]N)) & \text{by Lemma 15} \\ & \equiv & \mathrm{nf}_\beta([erase(M)/y]erase(N)) & \\ & =_\eta & x & \end{array}$$

Finally, by completeness of $Co^{\vdash}$ (Theorem 14), $x:\sigma \vdash_{co} \mathrm{nf}_\beta([M/y]N):\rho$. ∎

Notice that proving the admissibility of (cut) for the system $Co^{\vdash}$ is equivalent to proving a cut-elimination theorem for the extended system $Co^{\vdash}$+(cut).

As pointed out in the introduction, if we had taken (cut) as a primitive of the system, then we would have had to eliminate it in any case in order to prove coherence. Moreover, (cut) as primitive would imply that coercions could "compute on themselves" (even without inputs), whereas the coercions provided by $Co^{\vdash}$ are guaranteed to be in β-nf. Thus, they only transform an element of a type to a supertype without any other kind of computation.

The reader may wonder why such a simple proof is possible for a system which contains higher-order types and/or why we couldn't just derive it from the proofs of more expressive systems. It is known that impredicative second-order logic requires powerful tools to yield cut-elimination or normalization proofs ([Gir71, GLT89]). Yet, in [LMS96], we give another direct proof of cut-elimination for $Co^{\vdash}$ plus (cut) by a proof that does not rely on completeness: a non-obvious exercise. For the reader interested in the issue, we analyse here the key difficulties of a direct proof and some analogies and differences with respect to other calculi. We use [GLT89] as a reference.

For a first-order system, the proof of cut-elimination considered in Chapter 13 of [GLT89] is divided into two main parts: sections 13.1 and 13.2. The first part is the basis for an induction on the size of derivations: "cuts" are permuted with other rules and, when possible, they are moved up. Among the key cases, there are two crucial ones: cases 6 and 7 (or 8). Case 6 has an arrow as the cut formula, and the point there is that if one "swaps" the cut rule with the right and left-arrow rules, the cut rule is, unlike the preceding cases, NOT moved up, and a straightforward induction would then fail. This is the reason for introducing (in section 13.2) the notion of "degree" δ of a formula and using a combined induction on derivations AND degrees: in case 6, section 13.1, the degree of the cut formula does decrease (not the size of the derivation).

Fortunately, in the first-order case, the notion of degree of a formula does not present any problems (section 13.2): it is preserved under instantiation of a term variable by a term (e.g., $\delta(A[t/x]) = \delta(A)$). By this, the degree of the cut formulae in case 7 (and 8) of section 13.1 does not change when moving up the cut rule. And the combined induction goes through.

However, this form of combined induction cannot be used in the presence of higher-order (impredicative) formulae: no degree or measure on formulae is preserved by instantiation in general. Girard and Tait's proof by "candidates of reducibility" employs a powerful technique to overcome this crucial difficulty of impredicative systems. The heavy inductive loading used (conditions CR 1,2,3 of chapters 6 and 14) requires the intended set of terms (candidates of reducibility) to be closed under reductions and expansions. In our case of a direct proof [LMS96], the difficulties of case 6 (section 13.1 of [GLT89]) are easily handled: that case corresponds to $(\rightarrow)$ occurring simultaneously on the left and on the right, where an arrow formula is eliminated, by cut. This gives a symmetric situation and allows one to move up the last cut rule, in contrast to case 6 in [GLT89]. Thus, we do not need to introduce an induction on degrees of formulae, which would, in turn, cause problems in an impredicative system, like ours. Moreover, we can neither refer to nor use the candidates of reducibility, even though $Co^{\vdash}$ is a subsystem of System F: the terms of our system with the cut rule are not closed under β-expansions (CR3), as pointed out before Lemma 5. Note that, in general, cut-elimination cannot automatically be applied to subsystems: there are easy counter-examples.

In the present approach, we used a simpler technique. The linearity of terms gives an immediate normalization theorem; then the completeness result (Theorem 14: coercions are exactly those F terms that erase to the identity) allows us to avoid a step-by-step cut-elimination and refer to the "evaluation" (nf_β) of a term's erasure (cf. recent results on "normalization-by-evaluation" [BS91]).

6.1 Bicoercibility

Consider now the term model of $\mid Co^{\vdash} \mid$ of $Co^{\vdash}$, i.e., the structure whose objects are types and arrows are coercions. $\mid Co^{\vdash} \mid$ is a category. Indeed, by (ax), it contains all identities. By the transitivity of $\vdash_{co}$, coercions (arrows) compose: just observe that if

$$x:\sigma \vdash_{co} M : \tau \qquad \text{and} \qquad y:\tau \vdash_{co} N : \rho$$

then there exists P such that $x:\sigma \vdash_{co} P : \rho$ and P is unique by coherence. As for associativity, this is again given by coherence.

$\mid Co^{\vdash} \mid$ is even a partial order: by the corollary below, anti-symmetry of $\vdash_{co}$ is a consequence of coherence and transitivity of entailment. Define first the following relation of *bicoercibility* between types:

Definition (Bicoercibility) *Two types σ and τ are defined to be bicoercible, written $\sigma \overset{\cong}{_b} \tau$, iff $\sigma \vdash_{co} \tau$ and $\tau \vdash_{co} \sigma$.*

For example, $\tau \overset{\cong}{\underset{b}{}} \forall X.\tau$ for X not free in τ. Now, recall that, in a category, two objects A and B are *isomorphic*, $A \cong B$, if there are maps $f : A \rightarrow B$ and $g : B \rightarrow A$ such that $g \circ f = id$ and $f \circ g = id$. Thus, one can prove the following:

Corollary (Anti-symmetry) *If* $\sigma \overset{\cong}{\underset{b}{}} \tau$ *then* $\sigma \cong \tau$ *in* $|Co^{\vdash}|$.

Proof: By assumption, $x : \sigma \vdash_{co} M : \tau$ and $y : \tau \vdash_{co} N : \sigma$. By (cut), we obtain $x : \sigma \vdash_{co} \mathrm{nf}_\beta([M/y]N) : \sigma$ and $y : \tau \vdash_{co} \mathrm{nf}_\beta([N/x]M) : \tau$, then, by coherence, $\mathrm{nf}_\beta([M/y]N) =_{\eta co} x$ and $\mathrm{nf}_\beta([N/x]M) =_{\eta co} y$. ∎

Note that bicoercibility is strictly stronger than isomorphism: the type $\sigma \rightarrow (\tau \rightarrow \rho)$ is isomorphic to $\tau \rightarrow (\sigma \rightarrow \rho)$ (see [Sol83, BDL92] for a characterisation) but it is clearly not a subtype, and so not bicoercible. Tiuryn in [Tiu95] has shown that bicoercibility is decidable, while Tiuryn and Urzyczyn [TU96] have shown that coercibility $\vdash_{co}$, that is, subtyping, is undecidable.

As pointed out, $|Co^{\vdash}|$ is a category and a partial order. This allows a preliminary observation on adding base types (*int*, *real*, etc.) with axioms introducing $\vdash_{co}$ between these types (e.g., $int \vdash_{co} real$). In short, one obtains the freely generated partial order, from these base types, by our axioms and rules. A proof-theoretic analysis of this fact will be given in Section 8.

7 Mitchell's subtyping system

In [Mit88], a "retyping function" is defined as a typed term in System F whose erasure η-reduces to the identity. In [Mit88, Lemma 9], it is then shown that σ is a subtype of τ in all "simple inference models" (as defined in [Mit88, section 4.2]) if and only if there is a retyping function from σ to τ. Thus, our Theorem 14 also yields semantic completeness, in the sense of Mitchell, for $Co^{\vdash}$. In this section, we give a direct comparison of $Co^{\vdash}$ to Mitchell's axiomatic approach to subtyping, presented here in a revised (but clearly equivalent) way.

Mitchell's Subtyping System

(ax) $\sigma \leq \sigma$

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

($\rightarrow$) $$\frac{\sigma' \leq \sigma \qquad \tau \leq \tau'}{(\sigma \rightarrow \tau) \ \leq \ (\sigma' \rightarrow \tau')}$$

($\forall$ subst) $$\frac{\sigma \leq \tau}{\forall X.\sigma \leq \forall X.\tau}$$

($\forall$ intro) $\sigma \ \leq \ \forall X.\sigma$

* *for X not free in σ*

($\forall$ elim) $\forall X.\sigma \ \leq \ [\rho/X]\sigma$

($\forall\!\rightarrow$ distr) $\forall X.(\sigma \rightarrow \tau) \ \leq \ (\forall X.\sigma) \rightarrow (\forall X.\tau)$

It is not hard to show that $Co^{\vdash}$ and Mitchell's system are equivalent.

Theorem 17 *$\sigma \leq \tau$ iff $\sigma \vdash_{co} \tau$.*

Proof: Clearly, rules (ax) and ($\rightarrow$) are identical in the two systems (with $\vdash_{co}$ for $\leq$). For the implication from left to right, the rule (trans) in Mitchell's system above corresponds to the $Co^{\vdash}$ rule (cut) which, by Theorem 16, is admissible for $Co^{\vdash}$; the other cases in this direction are proven as follows:

Case: ($\forall$ intro) is derivable in $Co^{\vdash}$ since X is not free in σ

$$\frac{\sigma \vdash_{co} \sigma}{\sigma \vdash_{co} \forall X.\sigma}\ (\forall_0 \text{ right})$$

Case: ($\forall$ elim) is derivable in $Co^{\vdash}$

$$\frac{[\rho/X]\sigma \vdash_{co} [\rho/X]\sigma}{\forall X.\sigma \vdash_{co} [\rho/X]\sigma}\ (\forall \text{ left})$$

Case: ($\forall$ subst) is derivable in $Co^{\vdash}$

$$\cfrac{\cfrac{[X/X]\sigma \vdash_{co} \tau}{\forall X.\sigma \vdash_{co} \tau}\ (\forall \text{ left})}{\forall X.\sigma \vdash_{co} \forall X.\tau}\ (\forall_0 \text{ right})$$

Case: ($\forall\!\rightarrow$ distr) is derivable in $Co^{\vdash}$+(cut)

$$\cfrac{\cfrac{\forall X.(\sigma \rightarrow \tau) \vdash_{co} \sigma \rightarrow \tau \qquad \cfrac{\forall X.\sigma \vdash_{co} \sigma \qquad \tau \vdash_{co} \tau}{\sigma \rightarrow \tau \vdash_{co} (\forall X.\sigma) \rightarrow \tau}\ (\rightarrow)}{\forall X.(\sigma \rightarrow \tau) \vdash_{co+cut} (\forall X.\sigma) \rightarrow \tau}\ (\text{cut})}{\forall X.(\sigma \rightarrow \tau) \vdash_{co+cut} (\forall X.\sigma) \rightarrow (\forall X.\tau)}\ (\forall_1 \text{ right})$$

Conversely, the remaining cases of the implication from right to left are proven as follows:

Case: ($\forall$ left) is derivable in Mitchell's system by

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

using (trans) on ($\forall$ elim) and the premise of ($\forall$ left), i.e., $[\rho/X]\sigma \leq \tau$.

Case: ($\forall_n$ right) is derivable in Mitchell's system as follows:

$$\begin{array}{lll}
\sigma & \leq \forall X.\sigma & \text{by } (\forall \text{ intro}) \text{ and } (\forall_n \text{ right}) \text{ side-condition } X \text{ not free in } \sigma \\
 & \leq \forall X.(\tau_1 \to \ldots(\tau_n \to \tau)\ldots) & \text{by } (\forall \text{ subst}) \text{ on } (\forall_n \text{ right}) \text{ premise} \\
 & \leq (\forall X.\tau_1) \to \forall X.(\tau_2 \to \ldots(\tau_n \to \tau)\ldots) & \text{by } (\forall\!\to \text{distr}) \\
 & \leq (\forall X.\tau_1) \to ((\forall X.\tau_2) \to \forall X.(\tau_3 \to \ldots(\tau_n \to \tau)\ldots) & \text{by } (\forall\!\to \text{distr}) \text{ and } (\to) \\
 & \quad \vdots & \vdots \\
 & \leq (\forall X.\tau_1) \to ((\forall X.\tau_2) \to \ldots((\forall X.\tau_n) \to (\forall X.\tau)\ldots) & \\
 & & \text{by } (\forall\!\to \text{distr}) \text{ and } (\to) \\
 & \leq \tau_1 \to (\forall X.\tau_2) \to \ldots(\forall X.\tau_n) \to (\forall X.\tau)\ldots) & \\
 & & \text{by } (\to),\ (\forall \text{ intro}) \text{ and } (\forall_n \text{ right}) \text{ side-condition } X \text{ not free in } \tau_1 \\
 & \quad \vdots & \vdots \\
 & \leq \tau_1 \to (\tau_2 \to \ldots(\tau_n \to \forall X.\tau)\ldots) & \\
 & & \text{by } (\to),\ (\forall \text{ intro}) \text{ and } (\forall_n \text{ right}) \text{ side-condition } X \text{ not free in } \tau_n
\end{array}$$

∎

8 System $Co^{\vdash}$with base types: $Co^{\vdash}{+}B$

Consider now extending the language of $Co^{\vdash}$ with fresh type constants $\kappa_1, \kappa_2, \kappa_3$, ... For example, these could be base types such as *bool*, *int*, *real*. To assert that a subtyping relation holds between some of these base types, between κ_i and κ_j say, add a fresh term constant $c_{i,j}$ to the language and add the following Gentzen-style rule asserting that κ_i is a subtype of κ_j via coercion $c_{i,j}$:

$$(\kappa_i \leq \kappa_j) \quad \frac{x\!:\!\sigma \vdash M : \kappa_i}{x\!:\!\sigma \vdash c_{i,j}\, M : \kappa_j}$$

Let $Co^{\vdash}{+}B$ denote $Co^{\vdash}$ extended with such base types, terms and rules, and let $\vdash_{co+\mathrm{B}}$ denote entailment in this extended system. What happens then to the subtyping partial order and the coherence and transitivity properties of the "pure" calculus $Co^{\vdash}$?

First, observe that the expected subtyping judgment $x\!:\!\kappa_i \vdash_{co} c_{i,j}\, x : \kappa_j$ is easy to derive: just take $\sigma = \kappa_i$ and $M \equiv x$ in the above rule. Indeed, as we now show, the new constants and rules introduce no new subtyping judgments beyond those expected. In a sense, base coercions act like variables; they do not compute.

Lemma 18 (Conservativity of $Co^{\vdash}{+}B$) *Assume that types σ and τ do not contain occurrences of base types. If $\sigma \vdash_{co+\mathrm{B}} \tau$ then $\sigma \vdash_{co} \tau$.*

Proof: First, let α be a $Co^{\vdash}+B$ derivation of a sequent S, where S may contain base types. Observe that by uniformly substituting, in α, a fresh variable X for all base types $\kappa_1, \ldots \kappa_n$ and omitting the $(\kappa_i \leq \kappa_j)$ rules, we obtain a $Co^{\vdash}$ derivation of $[X/\kappa_1, \ldots, X/\kappa_n]S$. This is easily shown by induction on $Co^{\vdash}+B$ derivations. The lemma then holds as a corollary of this observation, for the case when S does not contain base types. ∎

We now prove some properties of $\vdash_{\!\!\overline{co}+\mathrm{B}}$ derivations. For this, we require notions of equality on $Co^{\vdash}+B$ derivations, defined similarly to the equalities on $Co^{\vdash}$ derivations (see section 3.3), and for which we will use the same symbols: $=$, $=_\eta$, $=_{\eta co}$. Recall, in particular, that two derivations are $=$ when the proof terms labeling their final sequents are syntactically identical.

Lemma 19 *A $(\kappa_i \leq \kappa_j)$ rule cannot be applied at any point after an application of either $(\forall_n$ right) or $(\rightarrow)$.*

Lemma 20 *A $(\kappa_i \leq \kappa_j)$ rule can be permuted with $(\forall$ left) in both directions:*

$$\cfrac{\cfrac{S_1}{S_2}\;(\kappa_i \leq \kappa_j)}{S_3}\;(\forall\ \textit{left}) \qquad = \qquad \cfrac{\cfrac{S_1}{S_2'}\;(\forall\ \textit{left})}{S_3}\;(\kappa_i \leq \kappa_j)$$

Lemma 21 *Assume $\sigma \vdash_{\!\!\overline{co}+\mathrm{B}} \tau$ with derivation α.*

1. *If τ does not contain either "$\rightarrow$" or "$\forall$", then α contains applications of only (ax), $(\kappa_i \leq \kappa_j)$, and ($\forall$ left).*
2. *If σ does not contain either "$\rightarrow$" or "$\forall$", then α contains applications of only (ax), $(\kappa_i \leq \kappa_j)$, and ($\forall_0$ right).*

Proof: By induction on the size of α in both cases. Note that, in both cases, the derivations have just one branch. ∎

Theorem 22 (Subtypes and supertypes of a base type)

1. *Let $\sigma \vdash_{\!\!\overline{co}+\mathrm{B}} \kappa$. Then, $\sigma = \forall X_l \ldots \forall X_1.\sigma'$, for some $l \geq 0$ and where σ' is either a base type κ' or a variable X_i for $i \in 1 \ldots l$.*
2. *Let $\kappa \vdash_{\!\!\overline{co}+\mathrm{B}} \tau$. Then, $\tau = \forall X_m \ldots \forall X_1.\kappa'$ for some base type κ' and some $m \geq 0$.*

Proof of 1. By Lemma 21 case 1, the derivation of $\sigma \vdash_{\!\!\overline{co}+\mathrm{B}} \kappa$ consists of applications of only (ax), $(\kappa_i \leq \kappa_j)$, and ($\forall$ left). The derivation is linear, not a tree. The applications of $(\kappa_i \leq \kappa_j)$ can be permuted by Lemma 20 with the applications of ($\forall$ left), if any. Furthermore, (ax) must be instantiated with κ'. Thus, an equal derivation of $\sigma \vdash_{\!\!\overline{co}+\mathrm{B}} \kappa$ may be constructed with the following structure:

$$
\begin{array}{c l}
\kappa' \vdash_{co+\mathrm{B}} \kappa' & (\mathrm{ax}) \\
\cline{1-1}
 & (\forall\ \mathrm{left}) \\
\vdots & \quad\vdots \\
\cline{1-1}
\forall X_l \ldots \forall X_1 . \sigma' \vdash_{co+\mathrm{B}} \kappa' & (\forall\ \mathrm{left}) \\
\cline{1-1}
\forall X_l \ldots \forall X_1 . \sigma' \vdash_{co+\mathrm{B}} \kappa_p & (\kappa' \leq \kappa_p) \\
\cline{1-1}
\vdots & \quad\vdots \\
\cline{1-1}
\forall X_l \ldots \forall X_1 . \sigma' \vdash_{co+\mathrm{B}} \kappa_q & \quad\vdots \\
\cline{1-1}
\forall X_l \ldots \forall X_1 . \sigma' \vdash_{co+\mathrm{B}} \kappa & (\kappa_q \leq \kappa)
\end{array}
$$

where $k \geq 0$ is the number of applications of $(\kappa_i \leq \kappa_j)$ rules, i.e., $(\kappa' \leq \kappa_p), \ldots, (\kappa_q \leq \kappa)$, and $l' \geq 0$ is the number of applications of $(\forall$ left$)$. Note that $l \leq l'$.

Proof of 2. By Lemma 21 case 2, the derivation of $\kappa \vdash_{co+\mathrm{B}} \tau$ can contain applications of only (ax), $(\kappa_i \leq \kappa_j)$, and $(\forall_0$ right$)$. By Lemma 19, all the applications of the $(\kappa_i \leq \kappa_j)$ rules must appear before those of $(\forall_0$ right$)$. Furthermore, (ax) must be instantiated with the base type κ. Hence, the derivation has the following structure:

$$
\begin{array}{c l}
\kappa \vdash_{co+\mathrm{B}} \kappa & (\mathrm{ax}) \\
\cline{1-1}
\kappa \vdash_{co+\mathrm{B}} \kappa_p & (\kappa \leq \kappa_p) \\
\cline{1-1}
\vdots & \quad\vdots \\
\cline{1-1}
\kappa \vdash_{co+\mathrm{B}} \kappa_q & \quad\vdots \\
\cline{1-1}
\kappa \vdash_{co+\mathrm{B}} \kappa' & (\kappa_q \leq \kappa') \\
\cline{1-1}
\kappa \vdash_{co+\mathrm{B}} \forall X_1 . \kappa' & (\forall_0\ \mathrm{right}) \\
\cline{1-1}
\vdots & \quad\vdots \\
\cline{1-1}
\kappa \vdash_{co+\mathrm{B}} \forall X_m \ldots \forall X_1 . \kappa' & (\forall_0\ \mathrm{right})
\end{array}
$$

where $k \geq 0$ is the number of applications of $(\kappa_i \leq \kappa_j)$ rules, i.e., $(\kappa \leq \kappa_p), \ldots, (\kappa_q \leq \kappa')$, and $m \geq 0$ is the number of applications of $(\forall_0$ right$)$. ∎

Corollary

1. *Let* $x : \sigma \vdash_{co+\mathrm{B}} M : \kappa$. *Then,* $M \equiv c_k(\ldots c_1(x\rho_{l'} \ldots \rho_1) \ldots)$
2. *Let* $x : \kappa \vdash_{co+\mathrm{B}} M : \tau$. *Then,* $M \equiv \lambda X_m \ldots \lambda X_1 . c_k(\ldots c_1 x) \ldots)$

Theorem 22 shows that adding extra base types and base coercions changes the subtyping partial order in a reasonable way: each base type has only itself, the empty type, or other base types, given by the extra subtyping rules, as subtypes of it (up to bicoercibility, of course). Indeed, for case 1 of Theorem 22, $\sigma \underset{b}{\overset{\cong}{\leq}} \kappa'$ or $\sigma \underset{b}{\overset{\cong}{\leq}} \forall X.X$, while for case 2, $\tau \underset{b}{\overset{\cong}{\leq}} \kappa'$.

8.1 Transitivity

In order to prove transitivity of $\vdash_{co+\mathrm{B}}$, we will require the analogues of certain properties of pure $\vdash_{co}$, in particular, a completeness theorem.

First though, observe that adding base types and terms (in the sense of this paper) along with $(\kappa_i \le \kappa_j)$ rules to pure system F makes perfect sense. Let us call this extended system $F{+}B$. With this extended system, it is then simple to extend the proofs of properties of $Co^{\vdash}$ to $Co^{\vdash}{+}B$. For example, in the proof of Lemma 5 for $Co^{\vdash}$, we could treat terms $x:\sigma \vdash_{co} M : \tau$ as system F terms in light of Lemma 2. Here we can apply a similar proof and treat terms $x:\sigma \vdash_{co+\mathrm{B}} M : \tau$ as system $F{+}B$ terms.

Lemma 23 *Assume that $x:\sigma \vdash_{co+\mathrm{B}} M : \tau$ and $y:\tau \vdash_{co+\mathrm{B}} N : \rho$. Then, there is a unique, finite path of β-reductions from $[M/y]N$. Consequently, there exists a unique β-nf of $[M/y]N$.*

Proof: Analogous to the proof of Lemma 5. ∎

The definition of erasure (see section 5) must also be extended to account for base coercions.

Definition (Erasure of base coercions) *$erase(c_{i,j}\ M) \equiv erase(M)$.*

Lemma 24 *If $x:\sigma \vdash_{co+\mathrm{B}} M:\tau$ then $erase(M) \longrightarrow_{\eta} x$.*

Theorem 25 (Completeness of $Co^{\vdash}{+}B$) *Let M be a term in β-nf such that $x : \sigma \vdash_{F+\mathrm{B}} M : \tau$ and $erase(M) \longrightarrow_{\eta} x$. Then, $x : \sigma \vdash_{co+\mathrm{B}} M : \tau$.*

Proof: Almost identical to the proof of completeness for the pure system $Co^{\vdash}$ (Theorem 14) except that here M, in $\beta\eta$-nf, has the following general structure:

$$M \;\equiv\; \lambda\vec{Y}_1\,.\,\lambda y_1\!:\!\tau_1\ \ldots\ \lambda\vec{Y}_k\,.\,\lambda y_k\!:\!\tau_k\,.\,\lambda\vec{Y}_{k+1}\ .\ c_1(\ldots(c_l(x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1}))\ldots)$$

In the case where $c_i,\ldots,c_l$ are actually present, the type σ of x has the following general structure:

$$\sigma \;=\; \forall\vec{X}_1.(\sigma_1 \rightarrow\ \ldots\ \forall\vec{X}_n.(\sigma_n \rightarrow \forall\vec{X}_{n+1}.\sigma')\ldots)$$

with $[\vec{\rho}_{n+1}/\vec{X}_{n+1}]\sigma' \;=\; \kappa$ for some base type κ. This means that either σ' is a variable X in $\vec{X}_{n+1}$ and that κ is some type in $\vec{\rho}_{n+1}$, or that σ' is κ itself.
Using this observation, it is then easy to construct a $Co^{\vdash}{+}B$ derivation similar to the $Co^{\vdash}$ derivation at the end of the proof of Theorem 14 but with $(\kappa_i \le \kappa_j)$ rules inserted at the appropriate places in order to introduce the base coercions $c_1,\ldots,c_l$.
∎

Lemma 26 *Assume that $x:\sigma \vdash_{co+\mathrm{B}} M : \tau$ and $y:\tau \vdash_{co+\mathrm{B}} N : \rho$. Then, $erase(nf_{\beta}([M/y]N)) =_{\eta} nf_{\beta}(erase([M/y]N))$.*

Proof: Analogous to the proof of Lemma 15. ∎

Theorem 27 (Transitivity of $\vdash_{co+\mathrm{B}}$) *(cut) is an admissible rule in $Co^{\vdash}{+}B$.*

Proof: Analogous to the proof of transitivity of $\vdash_{co}$ (Theorem 16). ∎

We have thus shown that adding extra base types and coercions preserves the transitivity of entailment, i.e., of subtyping.

Note that, if we had used axioms of the form $\kappa_i \vdash_{co+\mathrm{B}} \kappa_j$ to assert subtyping relations between base types instead of Gentzen-style rules ($\kappa_i \leq \kappa_j$), it would have been impossible to eliminate cuts. Notice also that, usually, in Gentzen-style systems such as ours, "right" rules are balanced by symmetric "left" rules. In the system $Co^{\vdash}\!+B$, though, only the rules

$$(\kappa_i \leq \kappa_j) \quad \frac{x\!:\!\sigma \vdash_{co+\mathrm{B}} M : \kappa_i}{x\!:\!\sigma \vdash_{co+\mathrm{B}} c_{i,j}\, M : \kappa_j}$$

are added to assert subtyping between base types. However, from the admissibility of (cut) (Theorem 27), we can deduce the admissibility of the "left" analogue of the $(\kappa_i \leq \kappa_j)$ rules, although only up to β-convertibility in the labeled system:

$$\begin{array}{c} (\kappa_i \leq \kappa_j \text{ left}) \\ {}^{*}\ \textit{for } P \textit{ the } \beta\textit{-nf} \\ \textit{of } [c_{i,j}\, y/x]M \end{array} \quad \frac{x\!:\!\kappa_j \vdash_{co+\mathrm{B}} M : \sigma}{y\!:\!\kappa_i \vdash_{co+\mathrm{B}} P : \sigma}$$

To see this, assume that $x\!:\!\kappa_j \vdash_{co+\mathrm{B}} M : \sigma$ has been proved. Construct the following derivation with (cut):

$$\begin{array}{r} (\kappa_i \leq \kappa_j) \\ (\text{cut}) \end{array} \frac{\dfrac{y\!:\!\kappa_i \vdash_{co+\mathrm{B}} y : \kappa_i}{y\!:\!\kappa_i \vdash_{co+\mathrm{B}} c_{i,j}\, y : \kappa_j} \qquad x\!:\!\kappa_j \vdash_{co+\mathrm{B}} M : \sigma}{y\!:\!\kappa_i \vdash_{co+\mathrm{B}} P : \sigma}$$

where P is the β-nf of $[c_{i,j}\, y/x]M$. Since (cut) is admissible, we are done.

What happens now to coherence of the system in the presence of base types?

8.2 Coherence

Suppose that base types $\kappa_1, \kappa_2, \kappa_3, \ldots$ are given together with some $(\kappa_i \leq \kappa_j)$ rules, as described previously, but that no other conditions are added; in particular, neither compositionality, nor unicity (coherence), nor associativity of base coercions are asserted.

Yet, by Lemma 18 (conservativity), we know that extending the pure calculus $Co^{\vdash}$ with new base types and rules yields no new coercions between "pure" types (i.e., types not containing base types). As regards compositionality, transitivity of $\vdash_{co+\mathrm{B}}$ (Theorem 27) guarantees that, once embedded in our Gentzen-style system, base coercions compose. Moreover, as pointed out previously, only types bicoercible to $\forall X.X$ and to the new base types may be freshly related to the base types by subtyping in the extended system $Co^{\vdash}+B$.

Thus, the questions of unicity (coherence) and associativity remain unsettled for the new base types (and their bicoercible images) in the extended system.

As a weak form of coherence, here is what (eq appl2 co) allows us to prove, without any further conditions or assumptions on base types and rules:

Theorem 28 (Weak coherence of $Co^{\vdash}+B$) *Assume that both $c_{i,j}$ and $d_{i,j}$ are constant coercions between base types κ_i and κ_j. If $y:\sigma \vdash_{co+\mathrm{B}} M:\kappa_i$, then $x:\forall X.\sigma \vdash_{co+\mathrm{B}} c_{i,j}([xX/y]M) =_{\eta co} d_{i,j}([xX/y]M) : \kappa_j$.*

Proof: The assumption means that the following two rules have been added:

$$\frac{x:\sigma \vdash_{co+\mathrm{B}} M:\kappa_i}{x:\sigma \vdash_{co+\mathrm{B}} c_{i,j}\ M:\kappa_j} \qquad \frac{x:\sigma \vdash_{co+\mathrm{B}} M:\kappa_i}{x:\sigma \vdash_{co+\mathrm{B}} d_{i,j}\ M:\kappa_j}$$

Construct then the following derivation:

$$\frac{\dfrac{y:\sigma \vdash_{co+\mathrm{B}} M:\kappa_i}{y:\sigma \vdash_{co+\mathrm{B}} c_{i,j}\,M:\kappa_j} \qquad \dfrac{y:\sigma \vdash_{co+\mathrm{B}} M:\kappa_i}{y:\sigma \vdash_{co+\mathrm{B}} d_{i,j}\,M:\kappa_j}}{x:\forall X.\sigma \vdash_{co+\mathrm{B}} c_{i,j}([xX/y]M) =_{\eta co} d_{i,j}([xX/y]M) : \kappa_j}\ \text{(eq appl2 co)}$$

∎

In particular, since $x:\forall X.\kappa_i \vdash_{co+\mathrm{B}} xX:\kappa_i$, this weak coherence theorem implies equality of the composition of $c_{i,j}$ and $d_{i,j}$ with the coercion xX: just take $\sigma = \kappa_i$ and $M \equiv y$. Clearly though, the theorem does not imply the equality of $c_{i,j}$ and $d_{i,j}$.

In order to prove full coherence for $Co^{\vdash}+B$, we need to force unicity of coercions (coherence) on base types. This may be done by adding a rule for equality on base types as follows.

Definition $=_{\eta co\mathrm{B}}$ *is the least equivalence relation generated by $=_{\eta co}$ plus the following rule:*

(eq base co)
$$\frac{x:\kappa_i \vdash_{co+\mathrm{B}} M:\kappa_j \qquad x:\kappa_i \vdash_{co+\mathrm{B}} N:\kappa_j}{x:\kappa_i \vdash_{co+\mathrm{B}} M =_{\eta co\mathrm{B}} N : \kappa_j}$$

Note that, by the corollary to Theorem 22, the terms M and N must have the following structure $c_i(\ldots(c_1 x)\ldots)$ and $c'_j(\ldots(c'_1 x)\ldots)$, respectively.

Theorem 29 (Coherence of $Co^{\vdash}+B$) *Let α_1 and α_2 be two derivations of $x:\sigma \vdash_{co+\mathrm{B}} M:\tau$ and $x:\sigma \vdash_{co+\mathrm{B}} N:\tau$ respectively. Then, $M =_{\eta co\mathrm{B}} N$.*

Proof: The proof is similar to the proof of coherence for the pure system $Co^{\vdash}$ (Theorem 12) but with an extra case. Assume that one of α_1, α_2 ends with a $(\kappa_i \leq \kappa_j)$ rule. Then, $\tau = \kappa$ for some base type κ. By Lemma 21 case 1, both α_1 and α_2 may contain applications of only (ax), $(\kappa_i \leq \kappa_j)$, and ($\forall$ left) rules. Apply

Lemma 20 to both derivations, permuting all applications of $(\kappa_i \leq \kappa_j)$ rules before those of $(\forall$ left), thus obtaining the following two derivations:

$$\begin{array}{cl}
\kappa' \vdash_{co+\mathrm{B}} \kappa' & (\mathrm{ax}) \\ \cline{1-1}
\kappa' \vdash_{co+\mathrm{B}} \kappa_p & (\kappa' \leq \kappa_p) \\ \cline{1-1}
\vdots & \vdots \\ \cline{1-1}
\kappa' \vdash_{co+\mathrm{B}} \kappa_q & \vdots \\ \cline{1-1}
x : \kappa' \vdash_{co+\mathrm{B}} M : \kappa & (\kappa_q \leq \kappa) \\ \cline{1-1}
\vdots & (\forall\ \mathrm{left}) \\
 & \vdots \\ \cline{1-1}
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa & (\forall\ \mathrm{left})
\end{array}
\qquad
\begin{array}{cl}
\kappa' \vdash_{co+\mathrm{B}} \kappa' & (\mathrm{ax}) \\ \cline{1-1}
\kappa' \vdash_{co+\mathrm{B}} \kappa_{p'} & (\kappa' \leq \kappa_{p'}) \\ \cline{1-1}
\vdots & \vdots \\ \cline{1-1}
\kappa' \vdash_{co+\mathrm{B}} \kappa_{q'} & \vdots \\ \cline{1-1}
x : \kappa' \vdash_{co+\mathrm{B}} N : \kappa & (\kappa_{q'} \leq \kappa) \\ \cline{1-1}
\vdots & (\forall\ \mathrm{left}) \\
 & \vdots \\ \cline{1-1}
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa & (\forall\ \mathrm{left})
\end{array}$$

Note that the applications of $(\forall$ left) are the same in both derivations, but the applications of $(\kappa_i \leq \kappa_j)$ rules, and their number, may differ.

Ignore now the applications of $(\forall$ left) and consider the (sub)derivations of $x : \kappa' \vdash_{co+\mathrm{B}} M : \kappa$ and $x : \kappa' \vdash_{co+\mathrm{B}} N : \kappa$ in each derivation. By simple application of the equality rule (eq base co) above, obtain $x : \kappa' \vdash_{co+\mathrm{B}} M =_{\eta co\mathrm{B}} N : \kappa$. Then, since $(\forall$ left) preserves equality of coercions (implied by (eq appl2 co); see remark section 3.3), we are done. ■

Corollary *Each $(\kappa_i \leq \kappa_j)$ rule preserves equality of coercions.*

Proof: Assume that a rule $(\kappa \leq \kappa')$ has been added to the system, and assume that the equality $x : \sigma \vdash_{co+\mathrm{B}} M =_{\eta co\mathrm{B}} N : \kappa$ has been derived.

The equality implies that $x : \sigma \vdash_{co+\mathrm{B}} M : \kappa$ and $x : \sigma \vdash_{co+\mathrm{B}} N : \kappa$. By application of the rule $(\kappa \leq \kappa')$, we obtain both $x : \sigma \vdash_{co+\mathrm{B}} c_{i,j}\, M : \kappa'$ and $x : \sigma \vdash_{co+\mathrm{B}} c_{i,j}\, N : \kappa'$ where $c_{i,j}$ is the base coercion given by the rule.

Use now the same proof technique as in Theorem 29 to derive $x : \sigma \vdash_{co+\mathrm{B}} c_{i,j}\, M =_{\eta co\mathrm{B}} c_{i,j}\, N : \kappa'$. ■

Coherence guarantees unicity of coercions on all types. As in the pure system $Co^{\vdash}$, it implies that bicoercible types are isomorphic. And, as in $Co^{\vdash}$, coherence implies the associativity of coercion composition, as given by transitivity of entailment.

9 Conclusions

The purpose of the calculus $Co^{\vdash}$ presented in this paper is to give a coherent logical meaning to the notion of subtyping: σ is a subtype of τ if σ implies (entails) τ. This meaning is the most obvious relation between logical implication and naive set-theoretic inclusion. The main advantage of our approach is that $Co^{\vdash}$ has a sound logical "status", independently of its intended meaning for subtyping. This is obtained by presenting entailment in the frame of a second-order sequent calculus, where quantification is introduced and eliminated by right and left rules. We could then state and prove relevant properties such as coherence, completeness and the admissibility of (cut), which is equivalent to a cut-elimination theorem.

It should be clear why we do not take the (cut) rule as part of the definition of our subtyping system. In order to obtain coherence, we would need to eliminate it anyway. And coherence is used to prove anti-symmetry. Moreover, without (cut), all our proof-terms (definable coercions) are in normal form, as only (cut) may introduce redexes, exactly as in the λ-calculus.

It may be fair to say that $Co^{\vdash}$ is a minimal meaningful system for implication (entailement) that also handles second order universal quantification. Indeed, what weaker but still meaningful computation is there than "take an input and transform it into an element of a larger type"? And, intuitionistically, logical implications are computations. $Co^{\vdash}$ characterizes the logical entailements which are coercions and explicitly used this characterization in the main results.

In the final section, we extended $Co^{\vdash}$ with base types. Completeness, transitivity and coherence hold in this extended calculus, which is conservative over $Co^{\vdash}$. In view of the work in [BCGS91, CMMS91, CG92], further extensions can be studied, in particular, with a Top type, records, variants and bounded quantification. Moreover, a joint system F+$Co^{\vdash}$ could also be relevant for investigating general polymorphic subtyping.

Acknowledgments

We would like to thank Jerzy Tiuryn for his close analysis and critical insight into our work. We also acknowledge the comments of an anonymous referee on a previous version of this paper.

Sergei Soloviev's work was carried out while he was employed by BRICS, the Danish National Research Council's center for computer science (Århus, Denmark); his work was funded by two European Community HCM grants: "Typed Lambda Calculus" (while visiting ENS, Paris) and "CLICS II". Giuseppe Longo's work was also supported by the HCM grant "Typed Lambda Calculus".

Appendix

The following rules complete the definition of $=_{\eta co}$ (section 3.3); they just say that rules ($\rightarrow$) and ($\forall_n$ right) preserve equality of coercions:

(eq $\rightarrow$)

$$\frac{x':\sigma' \vdash_{co} M =_{\eta co} M' : \sigma \qquad y:\tau \vdash_{co} N =_{\eta co} N' : \tau'}{x:\sigma\rightarrow\tau \vdash_{co} \lambda x':\sigma'.[xM/y]N =_{\eta co} \lambda x':\sigma'.[xM'/y]N' \;:\; \sigma'\rightarrow\tau'}$$

(eq $\forall_{0\leq k\leq n}$ right)
* *for* X *not free in* σ
nor in $\tau_1,\ldots,\tau_n$,
for M *not of the form* $\lambda y.M'$,
for $x_{k+1},\ldots,x_n$ *fresh*

$$\frac{\begin{array}{r} x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k.M \;=_{\eta co}\; \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k.N \\ :\tau_1\rightarrow\ldots(\tau_n\rightarrow\tau)\ldots) \end{array}}{\begin{array}{r} x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k\ldots\lambda x_n:\tau_n.\lambda X.Mx_{k+1}\ldots x_n \\ =_{\eta co}\; \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k\ldots\lambda x_n:\tau_n.\lambda X.Nx_{k+1}\ldots x_n \\ :\tau_1\rightarrow\ldots(\tau_n\rightarrow\forall X.\tau)\ldots) \end{array}}$$

(eq $\forall_{0\leq n<k}$ right)
* *for* X *not free in* σ
nor in $\tau_1,\ldots,\tau_n$,
for M *not of the form* $\lambda y.M'$
for $\tau\equiv\tau_{n+1}\rightarrow\ldots(\tau_k\rightarrow\tau')\ldots)$

$$\frac{\begin{array}{r} x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k.M \;=_{\eta co}\; \lambda x_1:\tau_1\ldots\lambda x_k:\tau_k.N \\ :\tau_1\rightarrow\ldots(\tau_n\rightarrow\tau)\ldots) \end{array}}{\begin{array}{r} x:\sigma \vdash_{co} \lambda x_1:\tau_1\ldots\lambda x_n:\tau_n.\lambda X.\lambda x_{n+1}:\tau_{n+1}\ldots\lambda x_k:\tau_k.M \\ =_{\eta co}\; \lambda x_1:\tau_1\ldots\lambda x_n:\tau_n\lambda X.\lambda x_{n+1}:\tau_{n+1}\ldots\lambda x_k:\tau_k.N \\ :\tau_1\rightarrow\ldots(\tau_n\rightarrow\forall X.\tau)\ldots) \end{array}}$$

References

[ACC93] M. Abadi, L. Cardelli and P.-L. Curien. Formal parametric polymorphism. *Theoretical Computer Science* 121, pages 9–58, 1993.

[BCGS91] V. Breazu-Tannen, T. Coquand, C.A. Gunter, and A. Scedrov. Inheritance as implicit coercion. *Information and Computation* 93, pages 172–221, 1991.

[BDL92] K. Bruce, R. Di Cosmo, and G. Longo. Provable isomorphisms of types. *Mathematical Structures in Computer Science* 2:2, pages 231–247, 1992.

[BL90] K. Bruce and G. Longo. A modest model of records, inheritance, and bounded quantification. *Information and Computation* 87, pages 196–240, 1990.

[BS91] U. Berger and H. Schwichtenberg. An inverse of the evaluation functional for typed λ-calculus. Proceedings of the *6th IEEE Symposium on Logic in Computer Science (LICS)*, 1991.

[CG92] P.-L. Curien and Giorgio Ghelli. Coherence of subsumption, minimum typing and type-checking in $F_{\leq}$. *Mathematical Structures in Computer Science* 2, pages 55–92, 1992.

[CL91] L. Cardelli and G. Longo. A semantic basis for Quest. *Journal of Functional Programming* 1, pages 417–458, 1991.

[CMMS91] L. Cardelli, S. Martini, J.C. Mitchell, and A. Scedrov. An extension of system F with subtyping. *Information and Computation* 94, pages 4–56, 1994. First appeared in the proceedings of the Conference on Theoretical Aspects of Computer Software (Sendai, Japan), T. Ito and R. Meyer, eds., Lecture Notes in Computer Science 526, pages 750–770, Springer-Verlag, 1991.

[CW85] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys* 17:4, pages 471–522, 1985.

[Gir71] J.-Y. Girard. Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. Proceedings of the *2nd Scandinavian Logic Symposium*, J.E. Fenstad, ed., pages 63–92, North-Holland, 1971.

[GLT89] J.-Y. Girard, Y. Lafont and P. Taylor. *Proofs and types.* Cambridge Tracts in Theoretical Computer Science 7, Cambridge University Press, 1989.

[Hyl82] J.M.E. Hyland. The effective topos. Proceedings of *The L.E.J. Brouwer Centenary Symposium*, A.S. Troelstra and D.S. van Dalen, eds., pages 165–216, North-Holland, 1982.

[Hyl88] J.M.E. Hyland. A small complete category. *Annals of Pure and Applied Logic* 40, pages 135–165, 1988.

[Jac95] B. Jacobs. Subtyping and bounded quantification from a fibred perspective. Proceedings of the *Conference on Mathematical Foundations of Programming Semantics* (New Orleans, U.S.A), 1995.

[Kle67] S.C. Kleene. *Mathematical logic*, Wiley, 1967.

[LM91] G. Longo and E. Moggi. Constructive natural deduction and its ω-set interpretation. *Mathematical Structures in Computer Science* 1:2, pages 215–253, 1991.

[LMS93] G. Longo, K. Milsted, and S. Soloviev. The genericity theorem and the notion of parametricity in the polymorphic λ-calculus. *Theoretical Computer Science* 121, pages 323–349, 1993.

[LMS96] G. Longo, K. Milsted, and S. Soloviev. A logic of subtyping. Available by anonymous ftp from ftp.ens.fr as
/pub/dmi/users/longo/logicSubtyping.ps.Z.

[MR92] Q. Ma and J.C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. Proceedings of the *Conference on Mathematical Foundations of Programming Semantics*, S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, eds., Lecture Notes in Computer Science 598, pages 1–40, Springer-Verlag, 1992.

[Mit88] J.C. Mitchell. Polymorphic type inference and containment. *Information and Computation* 76:2-3, pages 211–249, 1988. Also appeared in *Logical Foundations of Functional Programming*, G. Huet, ed., pages 153–193, Addison-Wesley, 1990.

[PS94] B. Pierce and M. Steffen. Higher-Order Subtyping. To appear in: *Theoretical Computer Science*, 1996. A preliminary version appeared in the proceedings of the *IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET)*, June 1994, and as University of Edinburgh technical report ECS-LFCS-94-280 and Universität Erlangen-Nürnberg Interner Bericht IMMD7-01/94, January 1994.

[Sol83] S. Soloviev. The category of finite sets and Cartesian Closed Categories. *Journal of Soviet Mathematics* 22:3, pages 1387–1400, 1983.

[Tiu95] J. Tiuryn. Equational axiomatization of bicoercibility for polymorphic types. Proceedings of the *15th Conference on the Foundations of Software Technology and Theoretical Computer Science*, P.S. Thiagarajan, ed., Lecture Notes in Computer Science 1026, pages 166–179, Springer-Verlag, 1995.

[Tiu96] J. Tiuryn. A sequent calculus for subtyping polymorphic types. Proceedings of the *Conference on Mathematical Foundations of Computer Science*, Springer-Verlag, 1996.

[TU96] J. Tiuryn and P. Urzyczyn. The subtyping problem for second-order types is undecidable. Proceedings of the *11th IEEE Symposium on Logic in Computer Science (LICS)*, 1996.