ECOLE NORMALE SUPERIEURE

A Logic of Subtyping

Giuseppe LONGO
Kathleen MILSTED
Sergei SOLOVIEV

LIENS - 95 - 31

Département de Mathématiques et Informatique

CNRS URA 1327

A Logic of Subtyping

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

LIENS - 95 - 31

October 1995

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

[**]SPIIRAN, 14th line, 39
199178 St Petersbourg Russia

Adresse électronique : soloviev@dmi.ens.fr

A Logic of Subtyping[*]

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 Gal. Leclerc
92131 Issy-les-Moulineaux
France
milsted@issy.cnet.fr

Sergei Soloviev
SPIIRAN, 14th line, 39
199178, St. Petersburg
Russia
gor@iias.spb.su or
soloviev@dmi.ens.fr

June 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) calculus of sequents 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, which stresses the power of sequents and Gentzen-style proof methods. Proof techniques based on *cut-elimination* will be at the core of our results.

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.

[*]A preliminary version, 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.

However, subtyping, in the presence of the functional arrow $\rightarrow$ (and second order universal quantification $\forall$) presents some problems. Indeed, in all functional approaches 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: can we 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 7.1). 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 programming languages, 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 (in bits) of the variable's contents, in which case the conversion must be 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. The conversions implied by such coercions are performed statically at compile-time, for type-checking, etc. Coercions are used in this sense in [Mit88] for example, where they are known as "retyping functions".

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. We claim that this is a crucial property for the partial order of subtyping (the "cut" rule corresponds to "transitivity") as much as it is fundamental in logic (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], sect.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) $\sigma \vdash \sigma$ (→) $\dfrac{\sigma' \vdash \sigma \qquad \tau \vdash \tau'}{\sigma \rightarrow \tau \vdash \sigma' \rightarrow \tau'}$

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

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) $\dfrac{\sigma \vdash \tau_1 \rightarrow \ldots(\tau_n \rightarrow \tau)\ldots)}{\sigma \vdash \tau_1 \rightarrow \ldots(\tau_n \rightarrow \forall X.\tau)\ldots)}$
* *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.

Remark 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. This will turn out to be important in our proof of cut-elimination.

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 a β-reductions. 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: our non-obvious result is that transitivity can be derived or, from a logical perspective, that our system, in the presence of the rules for universal quantification and extended with (cut), has the cut-elimination property.

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]. Transitivity will be essential to this result.

Coherence will mean that derivability of $\sigma \vdash \tau$ implies a *unique* coercion from σ to τ. One of its consequences will be anti-symmetry. Coherence will be easily shown in our 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:

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

**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:

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

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. This is read "term M is assigned type σ relative to environment Γ". 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*

Reduction of terms is defined as usual by the closure of the following rules:

(β_1) $\quad (\lambda x\!:\!\sigma.M)N \longrightarrow_{\beta 1} [N/x]M$ $\qquad$ (β_2) $\quad (\lambda X.M)\tau \longrightarrow_{\beta 2} [\tau/X]M$

(η_1) $\quad \lambda x\!:\!\sigma.Mx \longrightarrow_{\eta 1} M$ $\qquad$ * *for $x \notin FV(M)$*

(η_2) $\quad \lambda X.MX \longrightarrow_{\eta 2} M$ $\qquad$ * *for $X \notin FV(M)$*

Where $FV(M)$ is the collection of free type and term variables in 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 also write "nf" prefixed by a reduction relation to indicate normal form with respect to the relation.

Equality of terms is defined by the compatible, reflexive, transitive closure of $\longrightarrow_{\beta\eta}$. We write $M =_{\beta\eta} N$ for this equality, known as *$\beta\eta$-convertibility.* We reserve the notation $M \equiv N$ for syntactic identity of terms up to renaming of bound variables, i.e., α-equivalence. For types, equality, $\sigma = \tau$, is just syntactic identity up to renaming of bound variables.

3 Systems $Co^{\vdash}$ and $Co^{\vdash}$+(cut)

3.1 System $Co^{\vdash}$

Let us define a sequent calculus of subtyping, referred to as System $Co^{\vdash}$ (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 but, of course, the calculus may be considered independently of this interpretation by referring just to its logical significance.

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

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

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

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

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

* *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 $x : \sigma \vdash_{co} M : \tau$. This presentation makes explicit the coercion terms involved in each subtyping judgment. We will refer to the first presentation as the "unlabelled" system, to the second as the "labelled" system.

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

(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)
* *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

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

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

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

We shall shortly see that the two cases of ($\forall_n$ right) labeled with coercion terms are disjoint.[1] In the rest of the paper, we shall refer to judgments of either presentation (with/without coercion terms) 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.*

Example. In $Co^{\vdash}$, $\bot = \forall X.X$ (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) *If $x : \sigma \vdash_{co} M : \tau$, then the set of free term variables of M is exactly $\{x\}$, and x occurs exactly once and always at the leftmost position in M. Furthermore, each bound term variable occurs exactly once in 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 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 shows that the coercion terms involved in ($\forall_n$ right) are unambiguously defined since M, in the assumption, must be in β-nf. Note too that, though ($\rightarrow$) 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 $x : \sigma \vdash_F M : \tau$ and $M \longrightarrow_\eta M'$. Then $x : \sigma \vdash_{co} M : \tau$ iff $x : \sigma \vdash_{co} M' : \tau$.*

Clearly, $\vdash_{co}$ is strictly weaker than $\vdash_F$. 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 16 (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 System $Co^{\vdash}$+(cut)

We are going to show below that $\vdash_{co}$ is a transitive relation. That is, in the unlabelled system, the rule:

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

is admissible.

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

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

Note though that $[M/y]N$ does not need to be a coercion; in particular, it does not need to be in β-nf. However, without making use of Strong Normalization for System F, we can prove that:

Lemma 5 (Cut-rule coercions) *Consider the coercions $x\!:\!\sigma \vdash_{co} M : \tau$ and $y\!:\!\tau \vdash_{co} N : \rho$. Let n be the number of λs in $[M/y]N$. Then each β-reduction path from $[M/y]N$ has at most n steps and reduces to a unique term P in β-nf.*

Proof: By the linearity of coercions, the term $[M/y]N$ is linear and each β-reduction decreases the number of λs. ■

This lemma motivates the following alternative version of labelled (cut)-rule, where $\text{nf}([M/y]N)$ means β-nf of $[M/y]N$:

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

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

In section 6, the proof of admissibility of (cut) can be obtained from a cut-elimination theorem for $Co^{\vdash}$ extended with (cut). We will thus work with derivations in the extended system $Co^{\vdash}$+(cut), where we use $\vdash_{co+cut}$ for entailment. Clearly, a "cut-free" derivation in $Co^{\vdash}$+(cut) is just a derivation in $Co^{\vdash}$. Note, that by this approach to the cut-rule, the Subject Reduction Theorem trivially holds in $Co^{\vdash}$+(cut).

3.3 Equality in $Co^{\vdash}$ and $Co^{\vdash}$+(cut)

Equality of coercions is defined, essentially, by η-equality plus a coercion version of the $F_{\leq}$ rule (eq appl2) given in [CMMS91]. 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 terms) *$=_{\eta co}$ is the least equivalence relation generated by α-convertibility plus*

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

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

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

Remark that (eq appl2 co) implies ($\forall$ 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.

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

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 system $F_{\leq}$ using the corresponding "non-coercion" rule (eq appl2).

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

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

$$\dfrac{\dfrac{\dfrac{y_1:[X/X]X \;\vdash_{co}\; y_1 : X \qquad \dfrac{y:[X/X]X \;\vdash_{co}\; y : X}{y_2:[\bot/X]X \;\vdash_{co}\; y_2X : X}\,(\forall\text{ left})}{x:\bot \;\vdash_{co}\; xX \;=_{\eta co}\; x\bot X \;:\; X}\,(\text{eq appl2 co})}{x:\bot \;\vdash_{co}\; \lambda X.xX \;=_{\eta co}\; \lambda X.x\bot X \;:\; \bot}\,(\text{eq }\forall_0\text{ right})}{x:\bot \;\vdash_{co}\; x \;=_{\eta co}\; x\bot \;:\; \bot}\,(\text{eq }\eta)$$

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

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

Besides equality of terms, we shall need equality of derivations. To clarify the relationship between different systems, we introduce here several equality 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* $x:\sigma \vdash_{co} M =_{\eta co} N$.

In the unlabelled system, equality relations are defined via equality relations between corresponding derivations of the labelled system.

Equality of derivations in System $Co^{\vdash}$+(cut) are defined likewise. In our cut-elimination proof, we shall show that, for each derivation α of $x:\sigma \vdash_{co+cut} M:\tau$, there exists a derivation α' in $Co^{\vdash}$ with *the same* M so only $=$ will actually be needed there.

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

Clearly, the semantics of rules (eq $\rightarrow$) 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 σ 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).

4 Some special classes of derivations

4.1 Pure variable derivations

Definition (Pure variable derivation) *A pure variable derivation is a derivation such that no type variable occurs both free and bound in the same sequent (in the unlabelled and labelled systems $Co^{\vdash}$ and $Co^{\vdash}$+(cut)).*

This notion of pure variable derivations is comparable with that of Kleene's (see [Kle67]).

Lemma 6 *Every derivation in $Co^{\vdash}$ or in $Co^{\vdash}$+(cut) 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 ($\rightarrow$) and (cut), it is enough to use the induction hypothesis. ∎

Lemma 7 *Every derivation in $Co^{\vdash}$ or in $Co^{\vdash}$+(cut) 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:

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

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 ($\rightarrow$) as follows:

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

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".

4.2 Permutations of rules

The following lemma gives the rule pairs that can be permuted in a derivation of $Co^{\vdash}$, resulting in an equal derivation.

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

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

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

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 labelling 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.

In the next lemma we describe permutability of rules with (cut) (in $Co^{\vdash}$+(cut)).

Lemma 9 *(Rule permutations II) The following pairs of derivations are equal, up to $=$:*

1. *($\forall_n$ right) on the right permutes with (cut):*

$$\cfrac{\cfrac{\alpha_l}{S_1} \qquad \cfrac{\cfrac{\alpha_r}{S_2}}{S_3}\,(\forall_n\ \mathit{right})}{S_4}\,(\mathit{cut}) \quad = \quad \cfrac{\cfrac{\cfrac{\alpha_l}{S_1} \qquad \cfrac{\alpha_r}{S_2}}{S_3'}\,(\mathit{cut})}{S_4}\,(\forall_n\ \mathit{right})$$

2. (∀ left) on the left permutes with (cut):

$$\dfrac{\dfrac{\dfrac{\alpha_l}{S_1}}{S_3}\,(\forall\ \mathit{left}) \qquad \dfrac{\alpha_r}{S_2}}{S_4}\,(\mathit{cut}) \qquad = \qquad \dfrac{\dfrac{\dfrac{\alpha_l}{S_1} \qquad \dfrac{\alpha_r}{S_2}}{S_3'}\,(\mathit{cut})}{S_4}\,(\forall\ \mathit{left})$$

Recall that equality of derivations = means that the terms labelling the end-sequents are identical.

Other rule pairs cannot be permuted at all, either because the resulting derivation would end with a sequent containing totally different types, or because of structural mismatches in types. In particular, the following derivations ending with (cut) **cannot be permuted** in the same way as above because of structural mismatches in types in the premises:

- (→) on the left does **not** permute with (cut):

$$\dfrac{\dfrac{\dfrac{\alpha_{l1}}{\sigma' \vdash_{co} \sigma} \qquad \dfrac{\alpha_{l2}}{\tau \vdash_{co} \tau'}}{\sigma \to \tau \vdash_{co} \sigma' \to \tau'}\,(\to) \qquad \dfrac{\alpha_r}{\sigma' \to \tau' \vdash_{co} \rho}}{\sigma \to \tau \vdash_{co+cut} \rho}\,(\text{cut})$$

- (→) on the right does **not** permute with (cut):

$$\dfrac{\dfrac{\alpha_l}{\rho \vdash_{co} \sigma \to \tau} \qquad \dfrac{\dfrac{\alpha_{r1}}{\sigma' \vdash_{co} \sigma} \qquad \dfrac{\alpha_{r2}}{\tau \vdash_{co} \tau'}}{\sigma \to \tau \vdash_{co} \sigma' \to \tau'}\,(\to)}{\rho \vdash_{co+cut} \sigma' \to \tau'}\,(\text{cut})$$

- (∀ left) on the right does **not** permute with (cut):

$$\dfrac{\dfrac{\alpha_l}{\mu \vdash_{co} \forall X.\sigma} \qquad \dfrac{\dfrac{\alpha_r}{[\rho/X]\sigma \vdash_{co} \tau}}{\forall X.\sigma \vdash_{co} \tau}\,(\forall\ \text{left})}{\mu \vdash_{co+cut} \tau}\,(\text{cut})$$

- ($\forall_n$ right) on the left does **not** permute with (cut):

$$\dfrac{\dfrac{\dfrac{\alpha_l}{\sigma \vdash_{co} \tau_1 \to \ldots(\tau_n \to \tau)\ldots)}}{\sigma \vdash_{co} \tau_1 \to \ldots(\tau_n \to \forall X.\tau)\ldots)}\,(\forall_n\ \text{right}) \qquad \dfrac{\alpha_r}{\tau_1 \to \ldots(\tau_n \to \forall X.\tau)\ldots) \vdash_{co} \rho}}{\sigma \vdash_{co+cut} \rho}\,(\text{cut})$$

4.3 Atomic derivations

The following notion of "atomic" derivations is defined for System $Co^{\vdash}$ only.

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

Lemma 10 *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 11 *(Transforming atomic derivations) 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 12 *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 ($\rightarrow$), 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 $x:\sigma \vdash_{co} M:\tau_1 \rightarrow \forall X.\tau_2$. Apply then Lemma 11 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 11 to α_2 yielding a derivation $\alpha_2' = \alpha_2$ ending with ($\forall_n$ right). Take $\alpha_1' \equiv \alpha_1$. ∎

5 Coherence of $Co^{\vdash}$

By coherence, we mean that a coercion from type σ to type τ should be 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 $=_{\eta co}$.

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) is used.

Theorem 13 (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 10 and 12, 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 admissibility of (cut) in the next section 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.

6 Admissibility of Cut

In this section, we show that $\vdash_{co}$ is a transitive relation.

The reader may wonder why a full section is needed to prove "cut-elimination" for a *subsystem* of System F. The point is that normalization and cut-elimination results are not inherited by subsystems; we will discuss the issue more closely after the proof of the main theorem.

Theorem 14 (Cut-elimination) *Every derivation in $Co^{\vdash}$+(cut) is equal (=) to a cut-free derivation, i.e., a derivation in $Co^{\vdash}$.*

Proof: By induction on the size of the derivation. Note that derivations here are not atomic. We prove the theorem for derivations with one application of (cut) in the end. Cut-elimination then holds for derivations with arbitrary numbers of cuts by simply eliminating the cuts one by one beginning with uppermost cuts.
Consider then the first application of (cut) in a derivation:

$$\dfrac{\dfrac{\alpha_l}{S_L} \qquad \dfrac{\alpha_r}{S_R}}{S}\,(\text{cut})$$

The derivations of S_L and S_R are cut-free, i.e., α_l and α_r are $\vdash_{co}$ derivations, whereas S is a $\vdash_{co+cut}$ sequent. We proceed by case analysis of the last rule used in these derivations.

Case: S_R derived by (ax). The derivation of S looks like:

$$\dfrac{\dfrac{\alpha_l}{x\!:\!\sigma \vdash_{co} M : \tau} \qquad y\!:\!\tau \vdash_{co} y : \tau}{x\!:\!\sigma \vdash_{co+cut} [M/y]y : \tau}\,(\text{cut})$$

Clearly, this derivation is = to the derivation of the left premise, which is cut-free.

Case: S_L derived by (ax). The derivation of S looks like:

$$\dfrac{x\!:\!\sigma \vdash_{co} x : \sigma \qquad \dfrac{\alpha_r}{y\!:\!\sigma \vdash_{co} N : \rho}}{x\!:\!\sigma \vdash_{co+cut} [x/y]N : \rho}\,(\text{cut})$$

Take the derivation α_r of the right premise $S_R \equiv y:\sigma \vdash_{co} N:\rho$. Everywhere that y occurs, replace it by x. This can be done without fear of wrong variable capture because the derivation of S is a pure variable one: since x occurs free in S_L, it does not occur bound in S_R. The "renamed" derivation of S_R, which is cut-free, is then $=$ to the derivation of S.

<u>Case:</u> S_R derived by ($\forall_n$ right).
First, permute ($\forall_n$ right) on the right with (cut) by Lemma 9, case 1:

$$\dfrac{\dfrac{\alpha_l}{S_L} \qquad \dfrac{\dfrac{\alpha_r}{S_2}}{S_R}\,(\forall_n\ \text{right})}{S}\,(\text{cut}) \qquad = \qquad \dfrac{\dfrac{\dfrac{\alpha_l}{S_L} \qquad \dfrac{\alpha_r}{S_2}}{S_R'}\,(\text{cut})}{S}\,(\forall_n\ \text{right})$$

By induction, there exists an equal cut-free derivation of S_R', which replaces the one with (cut).

<u>Case:</u> S_L derived by ($\forall$ left).
First, permute ($\forall$ left) on the left with (cut) by Lemma 9, case 2:

$$\dfrac{\dfrac{\dfrac{\alpha_l}{S_1}}{S_L}\,(\forall\ \text{left}) \qquad \dfrac{\alpha_r}{S_R}}{S}\,(\text{cut}) \qquad = \qquad \dfrac{\dfrac{\dfrac{\alpha_l}{S_1} \qquad \dfrac{\alpha_r}{S_R}}{S_L'}\,(\text{cut})}{S}\,(\forall\ \text{left})$$

<u>Case:</u> S_L derived by ($\to$).
The cut formula is a $\to$ type so S_R was derived either by ($\forall_n$ right) or ($\to$). The first of these subcases has been treated already, by permuting the (cut).

Subcase: S_R derived by ($\to$).
This is a situation where (cut) cannot be permuted "as such" (see page 17). The derivation of S looks like:

$$\dfrac{\dfrac{\dfrac{\alpha_{l1}}{x':\sigma' \vdash_{co} M:\sigma} \quad \dfrac{\alpha_{l2}}{y:\tau \vdash_{co} N:\tau'}}{x:\sigma\to\tau \vdash_{co} P_1:\sigma'\to\tau'}\,(\to) \qquad \dfrac{\dfrac{\alpha_{r1}}{x'':\sigma'' \vdash_{co} M':\sigma'} \quad \dfrac{\alpha_{r2}}{y':\tau' \vdash_{co} N':\tau''}}{z:\sigma'\to\tau' \vdash_{co} P_2:\sigma''\to\tau''}\,(\to)}{x:\sigma\to\tau \vdash_{co+cut} P:\sigma''\to\tau''}\,(\text{cut})$$

$$\begin{array}{rcl} \text{where}\quad P_1 & \equiv & \lambda x':\sigma'.\,[xM/y]N \\ P_2 & \equiv & \lambda x'':\sigma''.\,[zM'/y']N' \\ P & \equiv & \text{nf}([P_1/z]P_2) \\ & \equiv & \text{nf}(\lambda x'':\sigma''.\,[(\lambda x':\sigma'.[xM/y]N)/z][zM'/y']N') \end{array}$$

First, rearrange the derivations of the premises to apply (cut) earlier and ($\to$) last:

$$\dfrac{\dfrac{\dfrac{\alpha_{r1}}{x'':\sigma'' \vdash_{co} M':\sigma'} \quad \dfrac{\alpha_{l1}}{x':\sigma' \vdash_{co} M:\sigma}}{x'':\sigma'' \vdash_{co+cut} \text{nf}([M'/x']M):\sigma}\,(\text{cut}) \qquad \dfrac{\dfrac{\alpha_{l2}}{y:\tau \vdash_{co} N:\tau'} \quad \dfrac{\alpha_{r2}}{y':\tau' \vdash_{co} N':\tau''}}{y:\tau \vdash_{co+cut} \text{nf}([N/y']N'):\tau''}\,(\text{cut})}{x:\sigma\to\tau \vdash_{co+cut} Q:\sigma''\to\tau''}\,(\to)$$

where $Q \equiv \lambda x'' \!:\! \sigma'' . [x(\mathrm{nf}([M'/x']M))/y]\,\mathrm{nf}([N/y']N')$.

By induction, there exist equal cut-free derivations of the premises to replace the ones with (cut). The labels of these cut-free derivations are the same coercions $\mathrm{nf}([M'/x']M)$ and $\mathrm{nf}([N/y']N')$. Since the ($\rightarrow$) rule preserves coercions, the term Q is a coercion and is in β-nf, i.e., $\mathrm{nf}(Q) \equiv Q$. Finally, we prove $P \equiv Q$ as follows. Note that we make use of Lemma 1, that coercions contain at most one free term variable, given by their environment.

$$\begin{array}{rll}
P & \equiv \mathrm{nf}(\lambda x'' \!:\! \sigma'' . [(\lambda x' \!:\! \sigma' . [xM/y]N)/z][zM'/y']N') & \\
& \equiv \mathrm{nf}(\lambda x'' \!:\! \sigma'' . [(\lambda x' \!:\! \sigma' . [xM/y]N)M'/y']N') & z \text{ not free in } M', N' \\
& \equiv \mathrm{nf}(\lambda x'' \!:\! \sigma'' . [[M'/x']([xM/y]N)/y']N') & \\
& & \beta\text{-reduction inside nf preserves } \beta\text{-nf} \\
& \equiv \mathrm{nf}(\lambda x'' \!:\! \sigma'' . [([x([M'/x']M)/y]N)/y']N') & x' \text{ not free in } N \\
& \equiv \mathrm{nf}(\lambda x'' \!:\! \sigma'' . [x([M'/x']M)/y]([N/y']N')) & y \text{ not free in } N' \\
& \equiv \mathrm{nf}(\lambda x'' \!:\! \sigma'' . [x(\mathrm{nf}([M'/x']M))/y]\,\mathrm{nf}([N/y']N')) & \\
& & \text{Strong Normalization} \\
& \equiv \mathrm{nf}(Q) & \\
& \equiv Q &
\end{array}$$

Case: S_L derived by ($\forall_n$ right).
This is another situation where (cut) cannot be permuted "as such" (see page 17). The derivation of S looks like:[2]

$$\dfrac{\dfrac{\dfrac{\alpha_l}{x \!:\! \sigma \vdash_{co} P' : \tau_1 \rightarrow \dots \tau_n \rightarrow \tau}}{x \!:\! \sigma \vdash_{co} P : \tau_1 \rightarrow \dots \tau_n \rightarrow \forall X.\tau}(\forall_n \text{ right}) \qquad \dfrac{\alpha_r}{z \!:\! \tau_1 \rightarrow \dots \tau_n \rightarrow \forall X.\tau \vdash_{co} Q : \rho}}{x \!:\! \sigma \vdash_{co+cut} R : \rho}(\text{cut})$$

where

$$\begin{array}{rll}
P' & \equiv \lambda x_1 \!:\! \tau_1 \dots x_k \!:\! \tau_k . M & \text{where } M \text{ does not start with } \lambda y \\
P & \equiv \begin{cases} \lambda x_1 \!:\! \tau_1 \dots \lambda x_k \!:\! \tau_k \dots \lambda x_n \!:\! \tau_n . \lambda X . M x_{k+1} \dots x_n & \text{for } 0 \le k \le n \\ \lambda x_1 \!:\! \tau_1 \dots \lambda x_n \!:\! \tau_n . \lambda X . \lambda x_{n+1} \!:\! \tau_{n+1} \dots \lambda x_k \!:\! \tau_k . M & \text{for } 0 \le n < k \end{cases} & \\
R & \equiv \mathrm{nf}([P/z]Q) &
\end{array}$$

and X not free in σ nor in $\tau_1, \dots, \tau_n$. Moreover, by the definition of pure variable derivation, X is also not free in ρ.

Proceed now by analysis of the cut-free right derivation:

$$\dfrac{\alpha_r}{z \!:\! \tau_1 \rightarrow \dots \tau_n \rightarrow \forall X.\tau \vdash_{co} Q : \rho}$$

Observe that ρ will have the following general form:

$$\rho \ \equiv \ \rho_1 \rightarrow \dots \rho_m \rightarrow \rho'$$

[2]In the proof of this case, we will minimise the number of brackets () by employing the standard convention that $\rightarrow$ associates to the right. Thus, for example, instead of writing $\tau_1 \rightarrow (\dots \rightarrow (\tau_n \rightarrow \tau) \dots)$ we will write $\tau_1 \rightarrow \dots \tau_n \rightarrow \tau$.

for some $0 \leq m$ and some types $\rho_1, \dots, \rho_m, \rho'$. It can be proven, by induction on n, that the derivation α_r satisfies at least one of the following cases:

Subcase 1: $\rho' \equiv \forall Y.\rho''$, and the "$\forall Y$" can be traced back to an application of ($\forall_{m-j}$ right) for some $0 \leq j \leq m$. The derivation α_r looks like:

$$\dfrac{\dfrac{\gamma_1}{y_1:\rho_1 \vdash_{co} M_1:\tau_1} \qquad \dfrac{\dfrac{\dfrac{\gamma_j}{y_j:\rho_j \vdash_{co} M_j:\tau_j} \qquad \dfrac{\dfrac{\gamma'}{\begin{array}{l} z_j:\tau_{j+1}\to\dots\tau_n\to\forall X.\tau \\ \quad \vdash_{co} Q'_j:\rho_{j+1}\to\dots\rho_m\to\rho'' \end{array}}}{\begin{array}{l} z_j:\tau_{j+1}\to\dots\tau_n\to\forall X.\tau \\ \quad \vdash_{co} Q_j:\rho_{j+1}\to\dots\rho_m\to\forall Y.\rho'' \end{array}}(\forall_{m-j}\text{ right})}{\vdots}(\to)}{\begin{array}{l} z_1:\tau_2\to\dots\tau_n\to\forall X.\tau \\ \quad \vdash_{co} Q_1:\rho_2\to\dots\rho_j\to\rho_{j+1}\to\dots\rho_m\to\forall Y.\rho'' \end{array}}(\to)}{z:\tau_1\to\dots\tau_n\to\forall X.\tau \vdash_{co} Q:\rho_1\to\dots\rho_j\to\rho_{j+1}\to\dots\rho_m\to\forall Y.\rho''}(\to)$$

Subcase 2: $m = n$ and, for some $0 \leq j \leq m$, $\rho_{j+1} \equiv \tau_{j+1}$, $\dots$, $\rho_n \equiv \tau_n$, $\rho' \equiv \forall X.\tau$, and the "$\forall X$" in $\forall X.\tau$ can be traced back to an occurrence of an axiom. The derivation α_r looks like:

$$\dfrac{\dfrac{\gamma_1}{y_1:\rho_1 \vdash_{co} M_1:\tau_1} \qquad \dfrac{\begin{array}{c}\dfrac{\dfrac{\gamma_j}{y_j:\rho_j \vdash_{co} M_j:\tau_j} \qquad \begin{array}{l} z_j:\tau_{j+1}\to\dots\tau_n\to\forall X.\tau \\ \quad \vdash_{co} z_j:\tau_{j+1}\to\dots\tau_n\to\forall X.\tau \end{array}}{\begin{array}{l} z_{j-1}:\tau_j\to\dots\tau_n\to\forall X.\tau \\ \quad \vdash_{co} Q_{j-1}:\rho_j\to\tau_{j+1}\to\dots\tau_n\to\forall X.\tau \end{array}}(\to) \\ \vdots \end{array}}{\begin{array}{l} z_1:\tau_2\to\dots\tau_n\to\forall X.\tau \\ \quad \vdash_{co} Q_1:\rho_2\to\dots\rho_j\to\tau_{j+1}\to\dots\tau_n\to\forall X.\tau \end{array}}(\to)}{z:\tau_1\to\dots\tau_n\to\forall X.\tau \vdash_{co} Q:\rho_1\to\dots\rho_j\to\tau_{j+1}\to\dots\tau_n\to\forall X.\tau}(\to)$$

$$\begin{array}{lll} \text{where} \quad Q_{j-1} & \equiv \lambda y_j:\rho_j\,.\,[(z_{j-1}\ M_j)/z_j]z_j & \equiv \quad \lambda y_j:\rho_j\,.\,z_{j-1}\ M_j \\ \qquad \vdots & & \\ \qquad Q_1 & \equiv \lambda y_2:\rho_2\,.\,[(z_1\ M_2)/z_2]Q_2 & \\ \qquad Q & \equiv \lambda y_1:\rho_1\,.\,[(z\ M_1)/z_1]Q_1 & \\ & \equiv \lambda y_1:\rho_1 \dots \lambda y_j:\rho_j\,.\,z\ M_1\ \dots\ M_j & \end{array}$$

The fact that each term variable $z_1, \dots, z_j$ occurs in a coercion term only once (Lemma 1) has been used.

Subcase 3: $n \leq m$, and the "$\forall X$" in $\forall X.\tau$ can be traced back to an application of ($\forall$ left). The derivation α_r looks like:

$$\dfrac{\dfrac{\gamma_1}{y_1 : \rho_1 \vdash_{co} M_1 : \tau_1} \qquad \dfrac{\begin{matrix}\dfrac{\dfrac{\gamma_n}{y_n : \rho_n \vdash_{co} M_n : \tau_n} \qquad \dfrac{\dfrac{\gamma'}{z'_n : [\mu/X]\tau \vdash_{co} Q'_n : \rho_{n+1} \rightarrow \ldots \rho_m \rightarrow \rho'}}{z_n : \forall X.\tau \vdash_{co} Q_n : \rho_{n+1} \rightarrow \ldots \rho_m \rightarrow \rho'}\,(\forall\text{ left})}{}\,(\rightarrow) \\ \vdots \end{matrix}}{\begin{matrix} z_1 : \tau_2 \rightarrow \ldots \tau_n \rightarrow \forall X.\tau \\ \vdash_{co} Q_1 : \rho_2 \rightarrow \ldots \rho_n \rightarrow \rho_{n+1} \rightarrow \ldots \rho_m \rightarrow \rho' \end{matrix}}\,(\rightarrow)}{z : \tau_1 \rightarrow \ldots \tau_n \rightarrow \forall X.\tau \vdash_{co} Q : \rho_1 \rightarrow \ldots \rho_n \rightarrow \rho_{n+1} \rightarrow \ldots \rho_m \rightarrow \rho'}\,(\rightarrow)$$

$$\begin{array}{lll}
\text{where} & Q_n & \equiv [(z_n\ \mu)/z'_n]Q'_n \\
& Q_{n-1} & \equiv \lambda y_n : \rho_n \,.\, [(z_{n-1}\ M_n)/z_n]Q_n \\
& \vdots & \\
& Q_1 & \equiv \lambda y_2 : \rho_2 \,.\, [(z_1\ M_2)/z_2]Q_2 \\
& Q & \equiv \lambda y_1 : \rho_1 \,.\, [(z\ M_1)/z_1]Q_1 \\
& & \equiv \lambda y_1 : \rho_1 \ldots \lambda y_n : \rho_n . [(z\ M_1\ \ldots\ M_n)/z_n]Q_n \\
& & \equiv \lambda y_1 : \rho_1 \ldots \lambda y_n : \rho_n \,.\, [(z\ M_1\ \ldots\ M_n\ \mu)/z'_n]Q'_n
\end{array}$$

Again, the fact that each term variable $z_1, \ldots, z_n$ occurs in a coercion term only once (Lemma 1) has been used.

Now, return to the main derivation with cut and proceed by case analysis according to the three subcases above for the right derivation.

Subcase 1: Apply Lemma 8, case 2, to the right derivation, permuting the ($\forall_{m-j}$ right) downwards with ($\rightarrow$) j times, so that it becomes ($\forall_m$ right) just before the (cut). By this same lemma, the terms labelling the end sequents in the two derivations are identical, i.e., Q. The modified derivation is then a case (S_R derived by ($\forall$ right)) that has been treated before: just permute the ($\forall_m$ right) with (cut) by Lemma 9, case 1 then apply the induction hypothesis to the upper derivation ending with (cut).

Subcase 2: In this case, we will transform both the left and right derivations α_l and α_r of the main derivation so as to apply (cut) in a derivation of lesser size.
First, in the right derivation α_r, replace the axiom:

$$z_j : \tau_{j+1} \rightarrow \ldots \tau_n \rightarrow \forall X.\tau \vdash_{co} z_j : \tau_{j+1} \rightarrow \ldots \tau_n \rightarrow \forall X.\tau$$

by:

$$z'_j : \tau_{j+1} \rightarrow \ldots \tau_n \rightarrow \tau \vdash_{co} z'_j : \tau_{j+1} \rightarrow \ldots \tau_n \rightarrow \tau$$

Keeping all other derivations the same, apply the same rules in the same order so that all occurrences of $\forall X.\tau$ are ultimately replaced by τ. The result is the derivation:

$$\frac{\alpha'_r}{z' : \tau_1 \to \ldots \tau_n \to \tau \vdash_{co} Q' : \rho_1 \to \ldots \rho_j \to \tau_{j+1} \to \ldots \tau_n \to \tau}$$

where $Q' \equiv \lambda y_1 : \rho_1 \ldots \lambda y_j : \rho_j . z'\ M_1\ \ldots\ M_j$.
Second, in the left derivation α_l, omit the last ($\forall_n$ right) thus obtaining the subderivation:

$$\frac{\alpha_l}{x : \sigma \vdash_{co} P' : \tau_1 \to \ldots \tau_n \to \tau}$$

where $P' \equiv \lambda x_1 : \tau_1 \ldots \lambda x_k : \tau_k . M$ for some M not starting with λy.

Now, apply (cut) to these two derivations followed by ($\forall_n$ right):

$$\cfrac{\cfrac{\cfrac{\alpha_l}{x : \sigma \vdash_{co} P' : \tau_1 \to \ldots \tau_n \to \tau} \qquad \cfrac{\alpha'_r}{\begin{array}{l} z' : \tau_1 \to \ldots \tau_n \to \tau \\ \quad \vdash_{co} Q' : \rho_1 \to \ldots \rho_j \to \tau_{j+1} \to \ldots \tau_n \to \tau \end{array}}}{x : \sigma \vdash_{co+cut} R' : \rho_1 \to \ldots \rho_j \to \tau_{j+1} \to \ldots \tau_n \to \tau}\,(\text{cut})}{x : \sigma \vdash_{co+cut} R'' : \rho_1 \to \ldots \rho_j \to \tau_{j+1} \to \ldots \tau_n \to \forall X . \tau}\,(\forall_n \text{ right})$$

where $R' \equiv \text{nf}([P'/z']Q')$.
Note that the side-conditions on applying ($\forall_n$ right) are satisfied since X is not free in σ nor in $\tau_{j+1}, \ldots, \tau_n$ by the application of ($\forall_n$ right) in the original derivation with cut, and X is not free in $\rho \equiv \rho_1 \to \ldots \rho_j \to \ldots \rho_n \to \forall X . \tau$ since the original derivation is a pure variable one.
Observe that the (cut) above occurs in a derivation of lesser size than in the original derivation. Hence, we can eliminate the cut by induction, obtaining a cut-free derivation with the same label R', which is thus a coercion. Since ($\forall_n$ right) preserves coercions, R'' is likewise a coercion.
We must now show that $R'' \equiv R$ where R is the term labelling the original derivation with (cut). Recall that R is given by:

$$\begin{array}{l} R \equiv \text{nf}([P/z]Q) \\ P \equiv \begin{cases} \lambda x_1 : \tau_1 \ldots \lambda x_k : \tau_k \ldots \lambda x_n : \tau_n . \lambda X . M x_{k+1} \ldots x_n & \text{for } k \leq n \\ \lambda x_1 : \tau_1 \ldots \lambda x_n : \tau_n . \lambda X . \lambda x_{n+1} : \tau_{n+1} \ldots \lambda x_k : \tau_k . M & \text{for } n < k \end{cases} \end{array}$$

and that Q is given by:

$$Q \equiv \lambda y_1 : \rho_1 \ldots \lambda y_j : \rho_j . z\ M_1\ \ldots\ M_j$$

Recall also that, for $i = 1 \ldots j$,

$$y_i : \rho_i \vdash_{co} M_i : \tau_i$$

Taking into account that $j \leq n$, we proceed now by the three cases $j \leq k \leq n$, $k < j \leq n$ and $j \leq n < k$. To save on space, we will not write the types of bound term variables, nor multiple term λs (thus writing, for example, just $\lambda x_1 \ldots x_n . M$ instead of $\lambda x_1 : \tau_1 \ldots \lambda x_n : \tau_n . M$).

• For $j \leq k \leq n$

$R \equiv \mathrm{nf}([P/z]Q)$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,((\lambda x_1 \ldots x_k \ldots x_n.\lambda X.M x_{k+1} \ldots x_n)\ M_1\ \ldots\ M_j))$
definition of Q, P
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,([M_j/x_j] \ldots [M_1/x_1](\lambda x_{j+1} \ldots x_n.\lambda X.M\, x_{k+1} \ldots x_n)))$
β-reduction inside nf
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.[M_j/x_j] \ldots [M_1/x_1](M\, x_{k+1} \ldots x_n))$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.([M_j/x_j] \ldots [M_1/x_1]M)\, x_{k+1} \ldots x_n)$
term substitution where $x_{k+1}, \ldots, x_n$ were fresh
$\quad \equiv \lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.\mathrm{nf}(([M_j/x_j] \ldots [M_1/x_1]M)\, x_{k+1} \ldots x_n)$

and

$R' \equiv \mathrm{nf}([P'/z']Q')$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,((\lambda x_1 \ldots x_k\,.\,M)\ M_1\ \ldots\ M_j))$ definition of Q', P'
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,x_{j+1} \ldots x_k\,.\,[M_j/x_j] \ldots [M_1/x_1]M)$
β-reduction inside nf
$\quad \equiv \lambda y_1 \ldots y_j\,.\,x_{j+1} \ldots x_k\,.\,\mathrm{nf}([M_j/x_j] \ldots [M_1/x_1]M)$

Recall that M does not begin with λy and, by Lemma 1, the free variable x is its leftmost variable. Thus, the inner nf term of R' does not begin with a λy. The application of ($\forall_n$ right) to R' is thus allowed and yields R'' as follows, which is equal to R:

$R'' \equiv \lambda y_1 \ldots y_j\,.\,x_{j+1} \ldots x_k \ldots x_n.\lambda X.$
$\qquad (\mathrm{nf}([M_j/x_j] \ldots [M_1/x_1]M))\ x_{k+1} \ldots x_n$
($\forall_n$ right) applied to R'
$\quad \equiv \lambda y_1 \ldots y_j\,.\,x_{j+1} \ldots x_k \ldots x_n.\lambda X.$
$\qquad \mathrm{nf}(([M_j/x_j] \ldots [M_1/x_1]M)\ x_{k+1} \ldots x_n)$
nf does not begin with λy
$\quad \equiv R$

• For $k < j \leq n$

$R \equiv \mathrm{nf}([P/z]Q)$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,((\lambda x_1 \ldots x_k \ldots x_n.\lambda X.M x_{k+1} \ldots x_n)\ M_1\ \ldots\ M_j))$
definition of Q, P
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,([M_j/x_j] \ldots [M_1/x_1](\lambda x_{j+1} \ldots x_n.\lambda X.M\, x_{k+1} \ldots x_n)))$
β-reduction inside nf
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.[M_j/x_j] \ldots [M_1/x_1](M\, x_{k+1} \ldots x_n))$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.([M_j/x_j] \ldots [M_1/x_1]M)\, M_{k+1} \ldots M_j\ x_{j+1} \ldots x_n)$
term substitution and by Lemma 1,
y_i is only free term variable in M_i, $i = 1 \ldots j$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.([M_k/x_k] \ldots [M_1/x_1]M)\, M_{k+1} \ldots M_j\ x_{j+1} \ldots x_n)$
$FV(M) = x, x_1 \ldots, x_k$
$\quad \equiv \lambda y_1 \ldots y_j.x_{j+1} \ldots x_n\,.\,\lambda X.\mathrm{nf}(([M_k/x_k] \ldots [M_1/x_1]M)\, M_{k+1} \ldots M_j\ x_{j+1} \ldots x_n)$

and

$R' \equiv \mathrm{nf}([P'/z']Q')$
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,((\lambda x_1 \ldots x_k\,.\,M)\ M_1\ \ldots\ M_j))$ definition of Q', P'
$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_j\,.\,([M_k/x_k] \ldots [M_1/x_1]M)\ M_{k+1} \ldots M_j)$
β-reduction inside nf
$\quad \equiv \lambda y_1 \ldots y_j\,.\,\mathrm{nf}(([M_k/x_k] \ldots [M_1/x_1]M)\ M_{k+1} \ldots M_j)$

As before, the inner nf term of R' does not begin with a λy. The application of ($\forall_n$ right) to R' is thus possible yielding R'', which is equal to R:

$$\begin{aligned} R'' &\equiv \lambda y_1 \ldots y_j \,.\, x_{j+1} \ldots x_n . \lambda X. \\ &\qquad \mathrm{nf}(([M_k/x_k] \ldots [M_1/x_1]M)\ M_{k+1} \ldots M_j)\ x_{j+1} \ldots x_n \\ &\equiv R \end{aligned}$$

- For $0 \leq j \leq n < k$

$$\begin{aligned} R &\equiv \mathrm{nf}([P/z]Q) \\ &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j \,.\, ((\lambda x_1 \ldots x_n . \lambda X . \lambda x_{n+1} \ldots x_k \,.\, M)\ M_1\ \ldots\ M_j)) \\ &\qquad\qquad\qquad\qquad \text{definition of } Q, P \\ &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j \,.\, ([M_j/x_j] \ldots [M_1/x_1](\lambda x_{j+1} \ldots x_n . \lambda X . \lambda x_{n+1} \ldots x_k \,.\, M))) \\ &\qquad\qquad\qquad\qquad \beta\text{-reduction inside nf} \\ &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j . x_{j+1} \ldots x_n . \lambda X . \lambda x_{n+1} \ldots x_k \,.\, [M_j/x_j] \ldots [M_1/x_1]M) \\ &\equiv \lambda y_1 \ldots y_j . x_{j+1} \ldots x_n . \lambda X . \lambda x_{n+1} \ldots x_k \,.\, \mathrm{nf}([M_j/x_j] \ldots [M_1/x_1]M) \end{aligned}$$

and

$$\begin{aligned} R' &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j \,.\, P'\ M_1\ \ldots\ M_j) \\ &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j \,.\, ((\lambda x_1 \ldots x_k \,.\, M)\ M_1\ \ldots\ M_j)) \qquad \text{definition of } P' \\ &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j \,.\, ([M_j/x_j] \ldots [M_1/x_1](\lambda x_{j+1} \ldots x_k . M))) \\ &\qquad\qquad\qquad\qquad \beta\text{-reduction inside nf} \\ &\equiv \mathrm{nf}(\lambda y_1 \ldots y_j . x_{j+1} \ldots x_k \,.\, ([M_j/x_j] \ldots [M_1/x_1]M)) \\ &\equiv \lambda y_1 \ldots y_j . x_{j+1} \ldots x_k \,.\, \mathrm{nf}([M_j/x_j] \ldots [M_1/x_1]M) \end{aligned}$$

As before, the inner nf term in R' does not begin with a λy. The application of ($\forall_n$ right) to R' is thus possible, and yields R'' as follows which is equal to R:

$$\begin{aligned} R'' &\equiv \lambda y_1 \ldots y_j . x_{j+1} \ldots x_n . \lambda X . \lambda x_{n+1} \ldots x_k \,.\, \mathrm{nf}([M_j/x_j] \ldots [M_1/x_1]M) \\ &\qquad\qquad\qquad\qquad (\forall_n \text{ right) applied to } R' \\ &\equiv R \end{aligned}$$

Subcase 3: As in the previous case, we will transform both the left and the right derivations of the main derivation so as to apply (cut) in a derivation of lesser size.

First, in the right derivation α_r, omit ($\forall$ left) but keep all subsequent rule applications the same. This results in the derivation:

$$\frac{\alpha'_r}{z' \!:\! \tau_1 \to \ldots \tau_n \to [\mu/X]\tau \ \vdash_{co}\ Q' \!:\! \rho}$$

where $Q' \equiv \lambda y_1 \!:\! \rho_1 \ldots \lambda y_n \!:\! \rho_n \,.\, [(z'\ M_1\ \ldots\ M_n)/z'_n]Q'_n$.

Second, in the left derivation α_l, omit the last ($\forall_n$ right) thus obtaining the subderivation:

$$\frac{\alpha_l}{x \!:\! \sigma \ \vdash_{co}\ P' \!:\! \tau_1 \to \ldots \tau_n \to \tau}$$

where $P' \equiv \lambda x_1 \!:\! \tau_1 \ldots \lambda x_k \!:\! \tau_k . M$ for some M not starting with λy. In this subderivation, uniformly substitute μ for X to obtain:

$$\frac{\alpha'_l}{x \!:\! \sigma \ \vdash_{co}\ [\mu/X]P' \!:\! \tau_1 \to \ldots \tau_n \to [\mu/X]\tau}$$

Remember that X is not free in σ nor in $\tau_1, \ldots, \tau_n$.
Now, apply (cut) to these two derivations:

$$\dfrac{\dfrac{\alpha'_l}{x:\sigma \vdash_{co} [\mu/X]P':\tau_1 \rightarrow \ldots \tau_n \rightarrow [\mu/X]\tau} \quad \dfrac{\alpha'_r}{z':\tau_1 \rightarrow \ldots \tau_n \rightarrow [\mu/X]\tau \vdash_{co} Q':\rho}}{x:\sigma \vdash_{co+cut} R':\rho}\text{(cut)}$$

where $R' \equiv \mathrm{nf}([([\mu/X]P')/z']Q')$.
Observe that the cut occurs in a derivation of lesser size than in the original derivation. Hence, eliminate the (cut) by induction, obtaining a cut-free derivation with the same label R' which is thus a coercion.
We must now show that $R' \equiv R$ where R is the term labelling the original derivation with (cut). Recall that R is given by:

$$R \equiv \mathrm{nf}([P/z]Q)$$
$$P \equiv \begin{cases} \lambda x_1:\tau_1 \ldots \lambda x_k:\tau_k \ldots \lambda x_n:\tau_n.\lambda X.M x_{k+1} \ldots x_n & \text{for } 0 \leq k \leq n \\ \lambda x_1:\tau_1 \ldots \lambda x_n:\tau_n.\lambda X.\lambda x_{n+1}:\tau_{n+1} \ldots \lambda x_k:\tau_k.M & \text{for } 0 \leq n < k \end{cases}$$

and that Q is given by:

$$Q \equiv \lambda y_1:\rho_1 \ldots \lambda y_n:\rho_n\,.\,[(z\ M_1\ \ldots\ M_n\ \mu)/z'_n]Q'_n$$

Recall also that, for $i = 1 \ldots n$,

$$y_i:\rho_i \vdash_{co} M_i:\tau_i$$

We proceed now by the two cases $k \leq n$ and $n < k$. And, as before, we save on space by not writing the types of bound term variables, nor multiple term λs.

• For $0 \leq k \leq n$

$R \equiv \mathrm{nf}([P/z]Q)$

$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_n\,.$
$\qquad [((\lambda x_1 \ldots x_k \ldots x_n.\lambda X.M x_{k+1} \ldots x_n)\ M_1 \ldots M_n\ \mu)/z'_n]Q'_n)$
definition of Q, P

$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_n\,.$
$\qquad [([\mu/X][M_n/x_n] \ldots [M_k/x_k] \ldots [M_1/x_1](M\ x_{k+1} \ldots x_n))/z'_n]Q'_n)$
β-reduction inside nf

$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_n\,.[([\mu/X](([M_n/x_n] \ldots [M_1/x_1]M)\ M_{k+1} \ldots M_n))/z'_n]Q'_n)$
term substitution and by Lemma 1,
y_i is only free term variable in M_i, $i = 1 \ldots n$

$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_n\,.[([\mu/X](([M_k/x_k] \ldots [M_1/x_1]M)\ M_{k+1} \ldots M_n))/z'_n]Q'_n)$
$FV(M) = x, x_1 \ldots x_k$

$\quad \equiv \mathrm{nf}(\lambda y_1 \ldots y_n\,.[((([M_k/x_k] \ldots [M_1/x_1][\mu/X]M)\ M_{k+1} \ldots M_n)/z'_n]Q'_n)$
X not free in $M_1, \ldots, M_n$ because ???

and

$[\mu/X]P' \equiv [\mu/X](\lambda x_1:\tau_1 \ldots \lambda x_k:\tau_k.M)$
$\qquad \equiv \lambda x_1:\tau_1 \ldots \lambda x_k:\tau_k\,.\,[\mu/X]M \qquad X$ not free in $\tau_1, \ldots, \tau_n$

and

$$
\begin{aligned}
R' &\equiv \mathrm{nf}([([\mu/X]P')/z']Q') \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,.\, [((\lambda x_1 \ldots x_k.[\mu/X]M)\ M_1\ \ldots\ M_n)/z'_n]Q'_n) \\
&\qquad\qquad\qquad\qquad \text{definition of } Q', [\mu/X]P' \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,.\, [(([M_k/x_k] \ldots [M_1/x_1][\mu/X]M)\ M_{k+1}\ \ldots\ M_n)/z'_n]Q'_n) \\
&\qquad\qquad\qquad\qquad \beta\text{-reduction inside nf} \\
&\equiv R
\end{aligned}
$$

- For $0 \leq n < k$

$$
\begin{aligned}
R &\equiv \mathrm{nf}([P/z]Q) \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,. \\
&\qquad [((\lambda x_1{:}\tau_1 \ldots \lambda x_n{:}\tau_n.\lambda X.\lambda x_{n+1}{:}\tau_{n+1} \ldots \lambda x_k{:}\tau_k.M) \\
&\qquad\ M_1 \ldots M_n\ \mu)/z'_n]Q'_n) \qquad \text{definition of } Q, P \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,. \\
&\qquad [([\mu/X][M_n/x_n] \ldots [M_1/x_1](\lambda x_{n+1}{:}\tau_{n+1} \ldots \lambda x_k{:}\tau_k.M))/z'_n]Q'_n) \\
&\qquad\qquad\qquad\qquad \beta\text{-reduction inside nf} \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,. \\
&\qquad [(\lambda x_{n+1}{:}[\mu/X]\tau_{n+1} \ldots \lambda x_k{:}[\mu/X]\tau_k \,. \\
&\qquad\ [M_n/x_n] \ldots [M_1/x_1][\mu/X]M)/z'_n]Q'_n) \\
&\qquad\qquad\qquad\qquad X \text{ not free in } M_1, \ldots M_n \text{ because ???}
\end{aligned}
$$

and

$$
\begin{aligned}
[\mu/X]P' &\equiv [\mu/X](\lambda x_1{:}\tau_1 \ldots \lambda x_n{:}\tau_n \ldots \lambda x_k{:}\tau_k.M) \\
&\equiv \lambda x_1{:}\tau_1 \ldots \lambda x_n{:}\tau_n.\lambda x_{n+1}{:}[\mu/X]\tau_{n+1} \ldots \lambda x_k{:}[\mu/X]\tau_k \,.\, [\mu/X]M \\
&\qquad\qquad\qquad\qquad X \text{ not free in } \tau_1, \ldots, \tau_n
\end{aligned}
$$

and

$$
\begin{aligned}
R' &\equiv \mathrm{nf}([([\mu/X]P')/z']Q') \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,. \\
&\qquad [((\lambda x_1{:}\tau_1 \ldots \lambda x_n{:}\tau_n.\lambda x_{n+1}{:}[\mu/X]\tau_{n+1} \ldots \lambda x_k{:}[\mu/X]\tau_k \,. \\
&\qquad\ [\mu/X]M)\ M_1 \ldots M_n)/z'_n]Q'_n) \qquad \text{definition of } Q', [\mu/X]P'' \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,. \\
&\qquad [([M_n/x_n] \ldots [M_1/x_1](\lambda x_{n+1}{:}[\mu/X]\tau_{n+1} \ldots \lambda x_k{:}[\mu/X]\tau_k \,. \\
&\qquad [\mu/X]M))/z'_n]Q'_n) \qquad \beta\text{-reduction inside nf} \\
&\equiv \mathrm{nf}(\lambda y_1 \ldots y_n \,. \\
&\qquad [(\lambda x_{n+1}{:}[\mu/X]\tau_{n+1} \ldots \lambda x_k{:}[\mu/X]\tau_k \,. \\
&\qquad\ [M_n/x_n] \ldots [M_1/x_1][\mu/X]M)/z'_n]Q'_n) \\
&\equiv R
\end{aligned}
$$

∎

6.1 Remarks on the proof of cut-elimination

We have thus proven the admissibility of (cut) for the system $Co^{\vdash}$ by 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 inductive proof, on the size of derivations, 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 require powerful tools to yield cut-elimination or normalization proofs ([Gir71, GLT89]). For the reader interested in this issue, let's analyse closely some analogies and differences with respect to other calculi. We use [GLT89] as a reference.

First, for a first-order system, the proof of cut-elimination considered in Chapter 13 of [GLT89] is divided into two main parts. The first part is the base for an induction on the size of derivations, like ours: 'cuts" are swapped 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). The first, case 6, is concerned with implication, as cut-formula. The point there is that, if one "swaps" the cut-rule with the two right and left-arrow rules, the cut-rule is NOT moved up, as in all the preceding cases, and a straightforward induction would then fail. This is the reason for introducing, in the second main part (sect. 13.2), the notion of "degree" for a formula and use a combined induction on derivations AND degrees: in case 6, sect. 13.1, the degree of cut-formulae does decrease (not the size of the derivation).

Fortunately, in the first-order case, the notion of degree for a formula doesn't present any problem (sect. 13.2): it is preserved under instantiation of a variable by a term (e.g., d(A[t/x]) = d(A)). By this, the degree of the cut formulae in case 7 (and 8) of sect. 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 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, the difficulties of case 6 (sect. 13.1 of [GLT89]) are rather easily handled: that case corresponds to our case of ($\rightarrow$) 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 couldn't even 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 (cut-rule coercions). Similarly, cut-elimination cannot automatically be applied to subsystems.

6.2 Bi-coercibility

Consider now the term model of $|\,Co^{\vdash}\,|$ of $Co^{\vdash}$, i.e., the structure whose objects are types and arrows are coercions. $|\,Co^{\vdash}\,|$ is a category. Indeed, by (ax), it contains all identities. By the admissibility of (cut), 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.

$|\,Co^{\vdash}\,|$ is even a partial order: by the corollary below, anti-symmetry of $\vdash_{co}$ is a consequence of coherence and cut-elimination. Define first the following relation of *bi-coercibility* between types:

Definition (Bi-coercibility) *Two types σ and τ are defined to be bi-coercible, written $\sigma \cong_b \tau$, iff $\sigma \vdash_{co} \tau$ and $\tau \vdash_{co} \sigma$.*

For example, $\tau \cong_b \forall X.\tau$ 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 $f\!:\!A \to B$ and $g\!:\!B \to A$ such that $g \circ f = id$ and $f \circ g = id$. Thus, one can prove the following:

Corollary (Anti-symmetry) *If $\sigma \cong_b \tau$ 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 $x\!:\!\sigma \vdash_{co} \mathrm{nf}([M/y]N) : \sigma$ and $y\!:\!\tau \vdash_{co} \mathrm{nf}([N/x]M) : \tau$, then, by coherence, $\mathrm{nf}([M/y]N) =_{\eta co} x$ and $\mathrm{nf}([N/x]M) =_{\eta co} y$. ■

Note that bi-coercibility is strictly stronger than isomorphism: the type $\sigma \to (\tau \to \rho)$ is isomorphic to $\tau \to (\sigma \to \rho)$ (see [Sol83, BDL92] for a characterisation) but it is clearly not a subtype, and so not bi-coercible.

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 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). This will be a key step in relating our system to that of Mitchell [Mit88] in the next section.

Lemma 15 (Coercions erase to identity) *If* $x\!:\!\sigma \vdash_{co} M\!:\!\tau$ *then* $erase(M) \longrightarrow_{\eta} x$

Conversely, any linear map in $\beta\eta$-nf whose erasure η-reduces to a term variable is a coercion. This gives a complete characterization of coercions. The proof proceeds by syntactic analysis of the normal form.

Theorem 16 (Completeness) *Let* M *be a term in* $\beta\eta$*-nf such that* $x\!:\!\sigma \vdash_F M : \tau$ *and* $erase(M) \longrightarrow_{\eta} x$*. Then,* $x\!:\!\sigma \vdash_{co} M : \tau$*.*

Proof: 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 satisfying the same assumptions, i.e., linear maps 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:

$$y_i\!:\!\tau_i \vdash_{co} M_i : [\vec{\rho}_1/\vec{X}_1,\; \ldots,\; \vec{\rho}_i/\vec{X}_i]\sigma_i \qquad (1)$$

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 complex derivations, 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).

$$\dfrac{\dfrac{z:\delta_{n+1}\sigma' \vdash_{co} z:\tau'}{x_{n+1}:\delta_n(\forall\vec{X}_{n+1}.\sigma') \vdash_{co} x_{n+1}\vec{\rho}_{n+1} : \tau'}\ \begin{array}{l}\text{(ax) by (3)}\\ (\forall\text{ left})\end{array}}{}$$

$$\dfrac{(1)\ \ y_n:\tau_n \vdash_{co} M_n : \delta_n\sigma_n \qquad x_{n+1}:\delta_n(\forall\vec{X}_{n+1}.\sigma') \vdash_{co} x_{n+1}\vec{\rho}_{n+1} : \tau'}{z_n:\delta_n(\sigma_n \to \forall\vec{X}_{n+1}.\sigma') \vdash_{co} \lambda y_n:\tau_n\,.\,z_n M_n\vec{\rho}_{n+1} : \tau_n \to \tau'}\ (\to)$$

$$\dfrac{z_n:\delta_n(\sigma_n \to \forall\vec{X}_{n+1}.\sigma') \vdash_{co} \lambda y_n:\tau_n\,.\,z_n M_n\vec{\rho}_{n+1} : \tau_n \to \tau'}{x_n:\delta_{n-1}(\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')) \vdash_{co} \lambda y_n:\tau_n\,.\,x_n\vec{\rho}_n M_n\vec{\rho}_{n+1} : \tau_n \to \tau'}\ (\forall\text{ left})$$

$$\dfrac{(1)\ \ y_{n-1}:\tau_{n-1} \vdash_{co} M_{n-1} : \delta_{n-1}\sigma_{n-1} \qquad x_n:\delta_{n-1}(\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')) \vdash_{co} \lambda y_n:\tau_n\,.\,x_n\vec{\rho}_n M_n\vec{\rho}_{n+1} : \tau_n \to \tau'}{\vdots}\ (\to)$$

$$\vdots \qquad\qquad (\forall\text{ left})$$

$$\dfrac{\vdots}{z_1:\delta_1(\sigma_1 \to \ldots\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')\ldots) \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\,.\,z_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \tau'}\ (\to)$$

$$\dfrac{z_1:\delta_1(\sigma_1 \to \ldots\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')\ldots) \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\,.\,z_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \tau'}{x:\forall\vec{X}_1.(\sigma_1 \to \ldots\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')\ldots) \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \tau'}\ (\forall\text{ left})$$

$$\dfrac{x:\forall\vec{X}_1.(\sigma_1 \to \ldots\forall\vec{X}_n.(\sigma_n \to \forall\vec{X}_{n+1}.\sigma')\ldots) \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \tau'}{x:\sigma \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\lambda\vec{Y}_{n+1}\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \forall\vec{Y}_{n+1}.\tau'}\ (\forall_{n+1}\text{ right) due to (2)}$$

$$\dfrac{x:\sigma \vdash_{co} \lambda y_1:\tau_1\ldots\lambda y_n:\tau_n\lambda\vec{Y}_{n+1}\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \tau_1 \to \ldots\tau_n \to \forall\vec{Y}_{n+1}.\tau'}{\vdots}\ (\forall_n\text{ right) due to (2)}$$

$$\dfrac{\vdots}{x:\sigma \vdash_{co} \lambda\vec{Y}_1\,.\,\lambda y_1:\tau_1\ \ldots\ \lambda\vec{Y}_n\,.\,\lambda y_n:\tau_n\,.\,\lambda\vec{Y}_{n+1}\,.\,x\vec{\rho}_1 M_1\ldots\vec{\rho}_n M_n\vec{\rho}_{n+1} \ :\ \forall\vec{Y}_1.(\tau_1 \to \ldots\forall\vec{Y}_n.(\tau_n \to \forall\vec{Y}_{n+1}.\tau')\ldots)}\ (\forall_0\text{ right) due to (2)}$$

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

7.1 Mitchell's System for Subtyping

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 Theorems 13 and 14 also give a semantic completeness theorem, in the sense of Mitchell, for $Co^{\vdash}$. We conclude this section with a direct comparison of $Co^{\vdash}$ to Mitchell's axiomatic approach to subtyping, which is given here in a revised (but clearly equivalent) way.

(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) above corresponds to (cut), which, by Theorem 14, 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 $Co^{\vdash}$

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

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

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

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:

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

∎

8 Adding base types to $Co^{\vdash}$

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 to the language and add the following Gentzen-style rule formalizing that κ_i is a subtype of κ_j via coercion c:

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

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

First, observe that the expected subtyping judgment $x : \kappa_i \vdash_{co} c\, x : \kappa_j$ 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, constant 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 a fresh variable X for all base types $\kappa_1, \ldots \kappa_n$ in α, 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. Then, the fact follows as a corollary of this observation, for the case when S does not contain base types. ∎

We now prove some properties of $\vdash_{co+\mathrm{B}}$ derivations. For this, we require notions of equality on $Co^{\vdash}+B$ derivations, defined similarly to the equalities on $Co^{\vdash}$ derivations (see section 5), and for which we will use the same symbols: $=$, $=_\eta$, $=_{\eta co}$. As a particular reminder, two derivations are $=$ when the proof terms labelling their end-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 *(Rule permutations III) A $(\kappa_i \leq \kappa_j)$ rule can be permuted with $(\forall$ left) in both directions:*

$$\cfrac{\cfrac{S_1}{S_2}\,(\kappa_i \leq \kappa_j)}{S_3}\,(\forall\ \mathit{left}) \quad = \quad \cfrac{\cfrac{S_1}{S_2'}\,(\forall\ \mathit{left})}{S_3}\,(\kappa_i \leq \kappa_j)$$

Proof: Easy. Note that the coercions are identical in the final sequents. ∎

Lemma 21 *Assume $\sigma \vdash_{co+\mathrm{B}} \tau$ 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 are linear, not trees. ∎

Theorem 22

1. *Let $\sigma \vdash_{co+\mathrm{B}} \kappa$. 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 $\kappa \vdash_{co+\mathrm{B}} \tau$. 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 $\sigma \vdash_{co+\mathrm{B}} \kappa$ 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 $\sigma \vdash_{co+\mathrm{B}} \kappa$ may be constructed with the following structure:

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

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:

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

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 bi-coercibility, of course). Indeed, for case 1 of Theorem 22, $\sigma \overset{\cong}{_b} \kappa'$ or $\sigma \overset{\cong}{_b} \forall X.X$, while for case 2, $\tau \overset{\cong}{_b} \kappa'$.

8.1 Admissibility of Cut for $Co^{\vdash}{+}B$

Let $Co^{\vdash}{+}B + (cut)$ denote $Co^{\vdash}{+}B$ extended with (cut). Following the paradigm used for the pure system $Co^{\vdash}$, we will show that (cut) is admissible for the extended system $Co^{\vdash}{+}B$ by proving cut-elimination for $Co^{\vdash}{+}B + (cut)$.

Lemma 23 *(Rule permutations IV) A $(\kappa_i \leq \kappa_j)$ rule can be permuted on the right with (cut). That is, the following derivation:*

$$\dfrac{x:\sigma \vdash_{co+B} M:\tau \qquad \dfrac{y:\tau \vdash_{co+B} N:\kappa_i}{y:\tau \vdash_{co+B} c\,N:\kappa_j}\,(\kappa_i \leq \kappa_j)}{x:\sigma \vdash_{co+B} P:\kappa_j}\,(cut)$$

where P is the β-nf of $[M/y](c\,N)$, is $=$ to:

$$\dfrac{\dfrac{x:\sigma \vdash_{co+B} M:\tau \qquad y:\tau \vdash_{co+B} N:\kappa_i}{x:\sigma \vdash_{co+B} Q:\kappa_i}\,(cut)}{x:\sigma \vdash_{co+B} c\,Q:\kappa_j}\,(\kappa_i \leq \kappa_j)$$

where Q is the β-nf of $[M/y]N$.

Proof: Easy. Observe that $P \equiv c\,Q$. ∎

Theorem 24 (Cut-elimination for $Co^{\vdash}{+}B + (cut)$) *Every derivation in $Co^{\vdash}{+}B + (cut)$ is $=$ to a derivation in $Co^{\vdash}{+}B$.*

Proof: By induction on the size of cut-free derivations (i.e., $Co^{\vdash}{+}B$ derivations). The proof is identical to the proof of Theorem 14 but with two extra cases. Consider thus the first application of (cut) in a derivation:

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

where the derivations of S_L and S_R are cut-free. The two extra cases to consider are as follows:

Case: S_R derived by a $(\kappa_i \leq \kappa_j)$ rule. Apply Lemma 23 to permute $(\kappa_i \leq \kappa_j)$ on the right with (cut) so that the (cut) moves upwards. Then, by induction, there exists an equal cut-free derivation, replacing the one with (cut).

Case: S_L derived by a $(\kappa_i \leq \kappa_j)$ rule. The derivation looks like:

$$\dfrac{\dfrac{\dfrac{\vdots}{\sigma \vdash_{co+B} \kappa_q}}{\sigma \vdash_{co+B} \kappa}\,(\kappa_q \leq \kappa) \qquad \dfrac{\vdots}{\kappa \vdash_{co+B} \tau}}{\sigma \vdash_{co+B} \tau}\,\text{(cut)}$$

Apply Theorem 22, case 1 to $S_L \equiv \sigma \vdash_{co+B} \kappa$, and case 2 to $S_R \equiv \kappa \vdash_{co+B} \tau$. This yields the following equal derivation, still with (cut).

$$
\dfrac{
\begin{array}{rc}
\begin{array}{r}(\text{ax})\\(\forall\ \text{left})\\ \vdots \\ (\forall\ \text{left})\\ (\kappa' \leq \kappa_p)\\ \vdots \\ \vdots \\ (\kappa_q \leq \kappa)\end{array}
&
\begin{array}{c}
\kappa' \vdash_{co+\mathrm{B}} \kappa' \\ \hline
\vdots \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa' \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa_p \\ \hline
\vdots \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa_q \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa
\end{array}
\end{array}
\qquad
\begin{array}{cl}
\begin{array}{c}
\kappa \vdash_{co+\mathrm{B}} \kappa \\ \hline
\kappa \vdash_{co+\mathrm{B}} \kappa_{p'} \\ \hline
\vdots \\ \hline
\kappa \vdash_{co+\mathrm{B}} \kappa_{q'} \\ \hline
\kappa \vdash_{co+\mathrm{B}} \kappa'' \\ \hline
\kappa \vdash_{co+\mathrm{B}} \forall X_1.\kappa'' \\ \hline
\vdots \\ \hline
\kappa \vdash_{co+\mathrm{B}} \forall X_m \ldots \forall X_1.\kappa''
\end{array}
&
\begin{array}{l}(\text{ax})\\(\kappa \leq \kappa_{p'})\\ \vdots \\ (\kappa_{q'} \leq \kappa'')\\ (\forall_0\ \text{right})\\ \vdots \\ (\forall_0\ \text{right})\end{array}
\end{array}
}{
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \forall X_m \ldots \forall X_1.\kappa''
}\ (\text{cut})
$$

Thus, $\tau = \forall X_m \ldots \forall X_1.\kappa''$ for m the number of applications of ($\forall_0$ right), and $\sigma = \forall X_l \ldots \forall X_1.\sigma'$ where σ' is either the base type κ' or a variable X_i for $i \in 1 \ldots l$, with $l \leq l'$ for l' the number of applications of ($\forall$ left).
Obtain now the following equal cut-free derivation by applying the rules in the derivation of S_R immediately after those of S_L:

$$
\begin{array}{cl}
\begin{array}{c}
\kappa' \vdash_{co+\mathrm{B}} \kappa' \\ \hline
\vdots \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa' \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa_p \\ \hline
\vdots \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa_q \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa_{p'} \\ \hline
\vdots \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa_{q'} \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \kappa'' \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \forall X_1.\kappa'' \\ \hline
\vdots \\ \hline
\forall X_l \ldots \forall X_1.\sigma' \vdash_{co+\mathrm{B}} \forall X_m \ldots \forall X_1.\kappa''
\end{array}
&
\begin{array}{l}(\text{ax})\\(\forall\ \text{left})\\ \vdots \\ (\forall\ \text{left})\\ (\kappa' \leq \kappa_p)\\ \vdots \\ \vdots \\ (\kappa_q \leq \kappa)\\ (\kappa \leq \kappa_{p'})\\ \vdots \\ (\kappa_{q'} \leq \kappa'')\\ (\forall_0\ \text{right})\\ \vdots \\ (\forall_0\ \text{right})\end{array}
\end{array}
$$

∎

We have thus shown that, in the extended system $Co^{\vdash}{+}B$, (cut) is admissible. In other words, 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. Remark 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

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

are added to assert subtyping between base types. However, from the admissibility of (cut) (Theorem 24), 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 labelled system:

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

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

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

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

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

8.2 Coherence of $Co^{\vdash}+B$

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; for example, no compositionality, no unicity (coherence), no associativity of these base coercions.

Yet, we know, by Lemma 18 (conservativity), 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, Theorem 24 guarantees that, once embedded in our Gentzen-style system, base coercions compose. Moreover, as pointed out previously, only types bi-coercible 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 bi-coercible 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 25 *Assume that both c and d are constant coercions between base types κ_i and κ_j. If $y{:}\sigma \vdash_{co+\mathrm{B}} M{:}\kappa_i$, then $x{:}\forall X.\sigma \vdash_{co+\mathrm{B}} c\,([xX/y]M) =_{\eta co} d\,([xX/y]M) : \kappa_j$*

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

$$\frac{x{:}\sigma \vdash_{co+\mathrm{B}} M : \kappa_i}{x{:}\sigma \vdash_{co+\mathrm{B}} c\ M : \kappa_j} \qquad \frac{x{:}\sigma \vdash_{co+\mathrm{B}} M : \kappa_i}{x{:}\sigma \vdash_{co+\mathrm{B}} d\ M : \kappa_j}$$

Construct then the following derivation:

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

∎

In particular, since $x{:}\forall X.\kappa_i \vdash_{co+\mathrm{B}} xX{:}\kappa_i$, Theorem 25 implies equality of the composition of c and d with the coercion xX: just take $\sigma = \kappa_i$ and $M \equiv y$. Clearly though, the Theorem does not imply the equality of c and d.

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:*

$$\text{(eq base co)}\quad \frac{x{:}\kappa_i \vdash_{co+\mathrm{B}} c_i(\ldots(c_1 x)\ldots) : \kappa_j \qquad x{:}\kappa_i \vdash_{co+\mathrm{B}} c'_j(\ldots(c'_1 x)\ldots) : \kappa_j}{x{:}\kappa_i \vdash_{co+\mathrm{B}} c_i(\ldots(c_1 x)\ldots) =_{\eta co\mathrm{B}} c'_j(\ldots(c'_1 x)\ldots)\ :\ \kappa_j}$$

Note that, by the corollary to Theorem 22, the terms $c_i(\ldots(c_1 x)\ldots)$ and $c'_j(\ldots(c'_1 x)\ldots)$ are the only ones possible.

Theorem 26 (Coherence of $Co^{\vdash}{+}B$ derivations) *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 13) 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:

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

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 $x : \sigma \vdash_{co+\mathrm{B}} c\, M : \kappa'$ and $x : \sigma \vdash_{co+\mathrm{B}} c\, N : \kappa'$ where c is the base coercion given by the rule.
Use now the same proof technique as in Theorem 26 to derive $x : \sigma \vdash_{co+\mathrm{B}} c\, M =_{\eta co\mathrm{B}} c\, N : \kappa'$. ∎

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

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. The main advantage of our approach is that $Co^{\vdash}$ has a sound logical "status", independently of its intended meaning for subtyping. This allowed us to (state and) prove relevant properties such as coherence and the admissibility of (cut), which is equivalent to a cut-elimination theorem. And, since Gentzen, cut-elimination theorems are at the core of (constructive) Proof Theory.

It should be clear why we do not take the (cut) rule as part of the definition of our "Logic of Subtyping", in spite of it being as fundamental as transitivity and it being required to prove completeness. 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) introduces redexes, exactly as in the lambda-calculus.

In view of its formal relation to (cut), it may be fair to say that this “Logic for Subtyping” is the least meaningful system for implication 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. By our system, we characterized the logical implications which are coercions and explicitly used this characterization in the main results.

Acknowledgments

We would like to thank Jerzy Tiuryn for his close analysis and critical insight into our work.

Sergei Soloviev's work was carried out while employed at BRICS, the Danish National Research Council's center for computer science (Århus, Denmark), and was funded by two European Community Human Capital and Mobility grants “Typed Lambda Calculus” (while visiting ENS, Paris) and “CLICS II”.

Appendix

The following rules complete the definition of $=_{\eta co}$ (Definition 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*

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

(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.

[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.

[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. To appear in the 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. To appear in the proceedings of the *11th IEEE Symposium on Logic in Computer Science (LICS)*, 1996.