Coercion completion and conservativity in coercive subtyping*

Sergei Soloviev
IRIT, Université Paul Sabatier
118 route de Narbonne
31062 Toulouse Cedex (France)
Email: soloviev@irit.fr

Zhaohui Luo
Department of Computer Science
University of Durham
South Road, Durham DH1 3LE, U.K.
Email: Zhaohui.Luo@durham.ac.uk

Abstract

Coercive subtyping offers a general approach to subtyping and inheritance by introducing a simple abbreviational mechanism to constructive type theories. In this paper, we study coercion completion in coercive subtyping and prove that the formal extension with coercive subtyping of a type theory such as Martin-Löf's type theory and UTT is a conservative extension. The importance of coherence conditions for the conservativity result is also discussed.

1 Introduction

One of important differences between type theory and set theory is that in the former we do not have a notion of subtype that corresponds to the notion of subset in the latter, though types often can be considered as inductively defined sets. The lack of useful subtyping mechanisms in dependent type theories with inductive types [CPM90, Dyb91, Luo94] and associated proof-development systems is a serious obstacle in their applications to large-scale formal development. Particularly, in the presence of inductive types which include types of natural numbers, lists, trees, and types of mathematical structures such as $\Sigma-$types, it is not clear how subtyping should be introduced to reason about subsets

*This work is supported partly by the project on Subtyping, Inheritance, and Reuse funded by UK EPSRC (GR/K79130, see `http://www.dur.ac.uk/CARG/sir.html`).

and represent inheritance, without compromising with good proof-theoretical properties.

The more traditional approach to subtyping considers usually a subtyping relation over lambda-terms. The notion of coercion was introduced later as an explicit representation of the transformation of (the elements of) the subtype into (the elements of) the supertype. The subtyping relation was interpreted by the existence of a certain definable term $c{:}A \to B$ when $A < B$, with motivation of giving semantics to calculi with subtyping and inheritance (see, e.g., [BCGS91], where no equational theory was studied for the calculus with coercions). Others have also considered coercions in different frameworks of subtyping. See, for example, [MCMS94, PS94, LMS95, Tiu95, Che96, Che98]. These studies had in common that coercion terms were supposed to possess a priori some special properties distinguishing them among all definable terms. For example, in [LMS95, Tiu95, Che96] coercions are "almost identical" maps. They are characterized formally as λ-terms that become η-equal in the untyped λ-calculus after all type information is erased.

Data types in dependent type theories such as Martin-Löf's type theory [NPS90] and the type theory UTT [Luo94], can in general be considered as inductive in the sense that they consist of their canonical objects. This is rather different from the traditional views when one studies type systems of programming languages and most of the work about subtyping, where objects constitute a pre-given universe, while types are assigned to the objects and a subtyping relation is obtained by overloading object terms (e.g. λ-terms). It is not clear (if possible) how the traditional approach to subtyping can be applied to type theory with inductive types in accordance with the view that types consist of canonical objects.

The framework on coercive subtyping [Luo97, Luo99] takes a new approach to subtyping and inheritance – taking coercions seriously and directly at the proof-theoretic level and providing a coherent view on how subtyping and inheritance can be studied in a type theory with inductive data types.

The basic idea of coercive subtyping in this context is that A is a subtype of B if there is a (unique) coercion c from A to B, and therefore, any object of type A may be regarded as an object of type B via c, where c is a functional operation from A to B in the type theory. Note that in our setting, coercions are *any* mapping between types, including those of user-defined coercions. We do not only consider those mappings with special properties (e.g., "almost identical"), but arbitrary mappings satisfying the uniqueness condition (more formally, the coherence condition).[1] In the theoretical framework of coercive subtyping, the role of c is represented by the following coercive definition rule:

$$(CD)\quad \frac{\Gamma \vdash f{:}\,(x{:}K)K' \quad \Gamma \vdash k_0{:}\,K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f(k_0) = f(c(k_0)){:}\,[c(k_0)/x]K'}$$

[1] The terminology 'coercion' used in this paper is quite different from some of the uses in the literature. For example, Chen [Che98] has used 'coercions' to represent 'almost identical' maps as mentioned above. We instead use the terminology in the general sense as given above.

which says that, if f is a functional operation with domain K, k_0 is an object of K_0, and c is a coercion from K_0 to K, then $f(k_0)$ is (well-typed and) definitionally equal to $f(c(k_0))$. Intuitively, we can view f as a context which requires an object of K; then the argument k_0 in the context f stands for its image of the coercion, $c(k_0)$. Therefore, one can use $f(k_0)$ as an abbreviation of $f(c(k_0))$.

The above simple idea, when formulated in a typed logical framework [Luo94], becomes very powerful. In our early work [Luo97, Luo99], we have developed the framework that covers subtyping relations represented by the following kinds of coercions:

- *Simple coercions*: representing subtyping between two types. For example, coercions between basic inductive types: the type of even numbers E is a subtype of the type of natural numbers N.

- *Parameterised coercions*: representing (point-wise) subtyping (or subfamily relation) between two families of types indexed by objects of the same type. A coercion can be parameterised over free variables occurring in it and (possibly) its domain or range types. As a special case, for example, each vector type $Vec(A, n)$ can be taken as a subtype of that of lists $List(A)$, parameterised by the index n, where the coercion would map the vector $< a_1, ..., a_n >$ to the list $[a_1, ..., a_n]$.

- *Coercions between parameterised inductive types*: we have general schematic rules that represent natural propagation of the basic coercions to other structured (or parameterised) inductive types. For example, $\Sigma(A, B)$ is a subtype of $\Sigma(A', B')$ if A is a subtype of A and B is a subfamily of B'.

Coercive subtyping has applications in many areas such as large proof development, inductive reasoning, representing implicit syntax (e.g., overloading), etc. See for example [Luo99] [2] .

Coercion mechanisms with certain restrictions have been implemented both in the proof development systems Lego [LP92] and Coq [Coq96], by Bailey [Bai96, Bai98] and Saibi [Sai97], respectively. Callaghan of the Computer Assisted Reasoning Group at Durham has recently implemented Plastic, a proof assistant that supports logical framework and coercive subtyping [Cal99], [CL00].

In the formal system of coercive subtyping, we distinguish basic and derived coercions. The system of basic coercions is open in the sense that new basic coercions may be declared (e.g., by the user). Though the new coercion terms specified as basic coercions need not have a priori special properties (they can be any functional maps), the set of basic coercions are required to be coherent (i.e., any two coercions of the same domain and range types are equal). Note that applications of the rules of logical framework (e.g., rules for dependent product introduction) permits to obtain derived coercions from the basic coercions.

[2] More recently, we have worked on *dependent coercions*, which have dependent product as kinds. These coercions correspond to subtyping between a type and a family of types (e.g., from lists to vectors). See [LS99] for details.

Even if the uniqueness of basic coercions is assured (the coherence condition), it doesn't imply in any direct way the uniqueness of derived coercions. In fact, that coercions between the same types are unique is closely related to the conservativity of the extension of the type theory with coercive subtyping. In this paper, we study the properties of general coercions and of type theory extended by coercive subtyping based on the properties of the system of basic coercions (such as coherence).

We shall focus on the issue of *coercion completion* and *conservativity*, one of the key meta-theoretic properties for coercive subtyping. The conservativity result says, intuitively, that every judgement that is derivable in the theory with coercive subtyping and that does not contain coercive applications is derivable in the original type theory.

Furthermore, for every derivation in the theory with coercive subtyping, one can always insert coercions correctly to obtain a derivation in the original type theory. Based on the meta-theoretic results presented in [JLS98], we show that the formal extension of coercive subtyping is a conservative extension of the original type theory, and hence justify that it is a truly abbreviational mechanism. The result is proved for a large class of basic subtyping rules, including those for parameterised inductive data types covered by inductive schemata, under the condition that the basic subtyping rules are coherent. We also discuss the importance of the coherence conditions (related to basic coercions) in the context of proving conservativity results.

The conservativity proof is difficult. It is partly due to the complexity of the dependent type system that is studied. More importantly, the difficulty comes from the fact that with coercive subtyping, it is necessary to consider coercion completion (the process of coercion insertion by means of an algorithm) based on the structure of derivations rather than terms or judgements. The main theorem, that the coercion completion map is total, shows that every derivation in the calculus extended by coercive subtyping can be transformed into a derivation in the original type theory (without subtyping). The result not only justifies the adequacy of abbreviational mechanism of coercive subtyping, but provides a further justification of the practical implementations of coercions in proof systems.

The proof of the conservativity theorem consists of the following three major parts:

1. Lemmas about general meta-theoretic properties of the theory with coercive subtyping;
2. Transitivity elimination in the calculus with subtyping and subkinding but without coercive application and definition rules.
3. The proof of the well-definedness (totality) of a coercion completion which maps derivations of the full theory into the calculus without coercive application and definition rules.

Most of the results about the structure of derivations in type theories with coercive subtyping (specified in typed logical framework) were presented in [JLS98],

as well as the theorem about transitivity elimination. New in this paper is the part about coercion completion and its relationship to conservativity. The main result is that coherence of basic subtyping rules does imply conservativity, under certain conditions (these conditions are satisfied, for example, for the type theory UTT or Martin-Löf's type theory.)

In Section 2, we give a formal description of the system with coercive subtyping, which is essentially the same as that presented in [Luo99], and introduce notations and general rule forms considered in the meta-theoretic development. Section 3 contains several basic meta-theoretic results of the calculus, about substitution, weakening, presupposed judgements and transitivity elimination. The issue of conservativity (and its relationship with coherence conditions) is discussed and explained in Section 4, where we also give an informal discussion on the main results and the methods used in the meta-theoretic development. Section 5 gives the definition of the coercion completion map and presents the main results about its properties. The conservativity results that follow from the results about coercion completion are presented in Section 6.

2 Coercive subtyping and subkinding

In this section, we give a formal presentation of the type theory extended with coercive subtyping.

2.1 Typed Logical Framework LF and Type Theories

The systems considered in this paper are extensions of the typed logical framework LF. LF is presented as in Chapter 9 of [Luo94] and in the paper [Luo99], with minor modifications. One of these modifications that should be noted is that we have included as part of the system several structural rules (weakening, context-retyping) that are usually treated as admissible. We need something more than mere admissibility: it is important that we have concrete elimination algorithms for these rules, substitutions and some other operations on derivations (extraction of derivations of presupposed judgements), and these algorithms are well-behaving w.r.t. coercion completion.

The inference rules of LF we use can be found in the appendix. For the reader less familiar with the notation used in the literature on logical frameworks, it may be noted that $[x{:}K]t$ represents abstraction (in terms) while $(x{:}K)K'$ is used for dependent product kinds. $(K)K'$ denotes dependent product $(x{:}K)K'$ when x doesn't occur free in K' and is sometimes denoted by $K \to K'$.

We consider a type theory T specified in LF and study its extensions with coercive subtyping. Examples of such type theories include Martin-Löf's intensional type theory [NPS90], UTT [Luo94], and many others.

These theories have in particular the means to define inductive data types. For example, in UTT, given any collection of inductive schemata $\overline{\Theta}$ in a context Γ, one may declare the constant expressions $\mathcal{M}[\overline{\Theta}]{:}\mathbf{Type}$ (inductive type itself), $\iota_i{:}\Theta_i[\mathcal{M}[\overline{\Theta}]]$ (introduction operators, corresponding to the schemata of

the collection $\overline{\Theta}$) and $E[\overline{\Theta}]$ (the elimination operator, which may be used to define functions with the domain $\mathcal{M}[\overline{\Theta}]$). Under such schemata, UTT contains a large class of inductive date types, including the type of natural numbers, the type of lists, types of trees, the type of ordinals, types of pairs (Σ-types), and types of dependent functions (Π-types). We give a simple example; for further details see [Luo94].

Example 2.1 *If $\overline{\Theta}$ is the sequence $X, (X)X$ then $\mathcal{M}[\overline{\Theta}]$ is the type of natural numbers. If we write $N \equiv \mathcal{M}[X, (X)X]$, there will be two introduction operators $\iota_0{:}N$ and $\iota_1{:}(N)N$ usually denoted by 0 and succ; the elimination operator Rec_N is the operator of primitive recursion (possibly with values in another type).*

Now, let T be any type theory specified in LF. We shall present the system $T[\mathcal{R}]$, the extension of T with coercive subtyping and subkinding, whose basic subtyping relation is given by the basic subtyping rules $\mathcal{R}$, which satisfy certain coherence conditions. In order to state the coherence conditions for the basic subtyping rules, we first consider an intermediate system $T[\mathcal{R}]_0$.

2.2 Coercive subtyping

The system $T[\mathcal{R}]_0$ spells out how subtyping relations are set up. $T[\mathcal{R}]_0$ is obtained from T by adding the subtyping judgement form $\Gamma \vdash A <_c B{:}\mathbf{Type}$, and the rules below. All coercions are functions, $\Gamma \vdash c{:}(El(A))El(B)$. The new rules of $T[\mathcal{R}]_0$ belong to the following groups:

- Basic subtyping rules.
- Transitivity and congruence rules for subtyping.
- Substitution rule for subtyping.

Basic subtyping rules. In general basic subtyping rules are the rules whose conclusions are subtyping judgements of the form $\Gamma \vdash A <_c B{:}\mathbf{Type}$. The premises may include some subtyping judgements of that form, and some judgements that do not contain subtyping or subkinding.

Example 2.2 *In coercive subtyping the coercion itself is part of the definition of a subtype. For example, one can define the type E of even numbers in the following way. Let N be the inductive type of natural numbers defined as above in Example 2.1. Define E as a 'copy' of N with $0_E{:}E$, $succ_E{:}E \to E$ and Rec_E, together with the coercion $E <_c N$:*

$$c \equiv Rec_E([x{:}E]N, 0, [y{:}N](succ(succ(y)))){:}E \to N$$

i.e., $c(0_E) = 0$ and $c(succ_E(e)) = succ(succ(c(e))$.

More sophisticated examples can be found in [Luo99], many of which come from the theory of inductive types [Luo94].

Transitivity and congruence rules.

$$(ST.2)\frac{\Gamma \vdash A <_c B{:}\mathbf{Type}\ \ \Gamma \vdash A = A'{:}\mathbf{Type}}{\Gamma \vdash A' <_c B{:}\mathbf{Type}}$$

$$(ST.3)\frac{\Gamma \vdash A <_c B{:}\mathbf{Type}\ \ \Gamma \vdash B = B'{:}\mathbf{Type}}{\Gamma \vdash A <_c B'{:}\mathbf{Type}}$$

$$(ST.4)\frac{\Gamma \vdash A <_{c_1} B{:}\mathbf{Type}\ \ \Gamma \vdash B <_{c_2} C{:}\mathbf{Type}}{\Gamma \vdash A <_{[x:El(A)]c_2(c_1(x))} C{:}\mathbf{Type}}$$

$$(ST.5)\frac{\Gamma \vdash A <_c B{:}\mathbf{Type}\ \ \Gamma \vdash c = c'{:}(El(A))El(B)}{\Gamma \vdash A <_{c'} B{:}\mathbf{Type}}$$

Substitution rule for subtyping

$$(ST.6)\frac{\Gamma, x{:}K, \Gamma' \vdash A <_c B{:}\mathbf{Type}\ \ \Gamma \vdash k{:}K}{\Gamma, [k/x']\Gamma' \vdash [k/x]A <_{[k/x]c} [k/x]B{:}\mathbf{Type}}$$

The following lemma is obvious, because the extension above has no means to obtain a judgement which is not of subtyping form from a subtyping judgement.

Lemma 2.3 *$T[\mathcal{R}]_0$ is a conservative extension of T, that is, if J is not of the form $A <_c B{:}\mathbf{Type}$, then $\Gamma \vdash J$ is derivable in T if and only if $\Gamma \vdash J$ is derivable in $T[\mathcal{R}]_0$.*

Coherence conditions

Definition 2.4 *(coherence conditions) We say that $\mathcal{R}$ is coherent if $T[\mathcal{R}]_0$ has the following properties:*

1. *$\Gamma \not\vdash A <_c A{:}\mathbf{Type}$ for any Γ, A and c.*
2. *If $\Gamma \vdash A <_c B{:}\mathbf{Type}$ and $\Gamma \vdash A <_{c'} B{:}\mathbf{Type}$, then $\Gamma \vdash c = c'{:}(El(A))El(B)$.*

In this paper, we assume that $\mathcal{R}$ is coherent.

Example 2.5 *Consider the types Rings, Groups and Monoids (they may be defined using Σ-types, cf. [Luo94]. One may define the coercions (using projections):*

$c_1{:}Rings \to Monoids$

$c_2{:}Rings \to Groups$

$c_3{:}Groups \to Monoids$

where c_1 maps a ring to its multiplicative monoid, and c_2 to its additive group; c_3 maps a group to its monoid. If $\mathcal{R}$ consists of any two of them, then it is coherent, but if include all three, the resulting system is not coherent since by c_1 we obtain the multiplicative monoid of a ring and by the composition of c_2 and c_3 its additive monoid.

2.3 Subkinding

A further intermediate system $T[\mathcal{R}]_{0K}$ is obtained from $T[\mathcal{R}]_0$ by adding the new subkinding judgement form $\Gamma \vdash K <_c K'$, and the following inference rules $(SK.1-9)$.

Basic subkinding rule

$$(SK.1)\frac{\Gamma\ \vdash A <_c B{:}\mathbf{Type}}{\Gamma \vdash El(A) <_c El(B)},$$

Subkinding for dependent products kinds

$$(SK.2)\frac{\Gamma \vdash K_1' <_{c_1} K_1 \quad \Gamma, x'{:}K_1' \vdash [c_1(x')/x]K_2 = K_2' \quad \Gamma, x{:}K_1 \vdash K_2\ \mathbf{kind}}{\Gamma \vdash (x{:}K_1)K_2 <_c (x'{:}K_1')K_2'},$$

where $c \equiv [f{:}(x{:}K_1)K_2][x'{:}K_1']f(c_1(x'))$;

$$(SK.3)\frac{\Gamma \vdash K_1' = K_1 \quad \Gamma, x{:}K_1' \vdash K_2 <_{c_2} K_2' \quad \Gamma, x{:}K_1 \vdash K_2\ \mathbf{kind}}{\Gamma \vdash (x{:}K_1)K_2 <_c (x{:}K_1')K_2'},$$

where $c \equiv [f{:}(x{:}K_1)K_2][x{:}K_1']c_2(f(x))$;

$$(SK.4)\frac{\Gamma \vdash K_1' <_{c_1} K_1 \quad \Gamma, x'{:}K_1' \vdash [c_1(x')/x]K_2 <_{c_2} K_2' \quad \Gamma, x{:}K_1 \vdash K_2\ \mathbf{kind}}{\Gamma \vdash (x{:}K_1)K_2 <_c (x'{:}K_1')K_2'},$$

where $c \equiv [f{:}(x{:}K_1)K_2][x'{:}K_1']c_2(f(c_1(x')))$.

Transitivity and congruence for subkinding.

$$(SK.5)\frac{\Gamma \vdash K_1 <_c K_2 \quad \Gamma \vdash K_2 = K_2'}{\Gamma \vdash K_1' <_{c'} K_2'}$$

$$(SK.6)\frac{\Gamma \vdash K_1 <_c K_2 \quad \Gamma \vdash K_1 = K_1'}{\Gamma \vdash K_1' <_{c'} K_2'} \quad (SK.7)\frac{\Gamma \vdash K <_{c_1} K' \quad \Gamma \vdash K' <_{c_2} K''}{\Gamma \vdash K <_{[x:K]c_2(c_1(x))} K''}$$

$$(SK.8)\frac{\Gamma \vdash K <_c K' \quad \Gamma \vdash c = c'{:}(K)K'}{\Gamma \vdash K <_{c'} K'}$$

Substitution rule for subkinding

$$(SK.9)\frac{\Gamma, x{:}K, \Gamma' \vdash K_1 <_c K_2 \quad \Gamma \vdash k{:}K}{\Gamma, [k/x]\Gamma' \vdash [k/x]K_1 <_{[k/x]c} [k/x]K_2}$$

Lemma 2.6 *$T[\mathcal{R}]_{0K}$ is a conservative extension of T and $T[\mathcal{R}]_0$.*

2.4 Coercive rules

The extension of T with coercive subtyping, the system $T[\mathcal{R}]$, is obtained from $T[\mathcal{R}]_{0K}$ by adding the following rules, which establish the essential connection between the original system T and its subtyping/subkinding extension.

Coercive application rules

$$(CA.1)\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k{:}K_0 \quad \Gamma \vdash K_0{<_c}K}{\Gamma \vdash f(k){:}[c(k)/x]K'}$$

$$(CA.2)\frac{\Gamma \vdash f = f'{:}(x{:}K)K' \quad \Gamma \vdash k_1 = k_2{:}K_0 \quad \Gamma \vdash K_0{<_c}K}{\Gamma \vdash f(k_1) = f'(k_2){:}[c(k_1)/x]K'}$$

Coercive definition rule

$$(CD)\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k_0{:}K_0 \quad \Gamma \vdash K_0{<_c}K}{\Gamma \vdash f(k_0) = f(c(k_0)){:}[c(k_0)/x]K'}$$

3 Basic meta-theoretic properties and transitivity elimination

3.1 Basic meta-theoretic properties.

As the technical background, we use several results from our paper [JLS98].

Elimination of wkn, retyping and substitutions.

Let $T[\mathcal{R}]^-$, $T[\mathcal{R}]^-_0$, $T[\mathcal{R}]^-_{0K}$, LF^- denote the corresponding system where the following rules have been removed: weakening wkn; context-retyping 3.3; the substitution rules of LF (rules 6.1-6.7); the substitution rules for subtyping and subkinding (rules ST.6, SK.9). (For the rule references, see Appendix.)

Theorem 3.1 *There is an algorithm* **E** *that transform every derivation d in $T[\mathcal{R}]$ into a derivation of the same judgement in $T[\mathcal{R}]^-$.*

Presupposed judgements and split-theorem.

In the split-theorem below, the so-called presupposed judgements and their derivations are considered. The notion of presupposed judgement is important for dependent type systems. Different from simple typed systems, the notions of well-formed type/context etc cannot be separated from that of the derivability in the system. Thus, for example, the judgement $\Gamma \vdash t{:}K$ *presupposes* the judgements $\Gamma \vdash$ **valid** (validity of the context) and $\Gamma \vdash K$**kind** (K is Γ-kind). The judgement $\Gamma \vdash El(A)$**kind** presupposes the judgement $\Gamma \vdash A{:}$**Type**.

In our system with subtyping, we have more judgement forms and more forms of presupposed judgements corresponding to them. For example, the judgement $\Gamma \vdash A <_c B{:}$**Type** has as presupposed judgements $\Gamma \vdash A{:}$**Type**, $\Gamma \vdash B{:}$**Type** and $\Gamma \vdash c{:}(El(A))El(B)$. When product kinds $\Gamma \vdash (x{:}K)K'$ are considered $\Gamma \vdash K$**kind** and $\Gamma, x{:}K \vdash K'$**kind** may be regarded as presupposed judgements as well.

Most of the forms of presupposed judgements may be found in the theorem below which we call "split-theorem". We shall not go into the formal details here since not all the proofs in this paper are presented in full detail.

Theorem 3.2 *(Split-theorem) There are algorithms* $\mathbf{v}$, $\mathbf{k}$, $\mathbf{p}_-{}^\circ$, $\mathbf{l}_=, \mathbf{r}_=$, $l^\circ_=, r^\circ_=$, $\mathbf{kd}$, $\mathbf{l}_<, \mathbf{r}_<, \mathbf{co}$ *that transform every* $T[\mathcal{R}]^-$*-derivation* d*,*

1. $\Gamma_1, \Gamma_2 \overset{d}{\vdash} J$ *into its subderivation* $\Gamma_1 \overset{\mathbf{v}_{\Gamma_1} d}{\vdash} \mathbf{valid}$*;*
2. $\Gamma_1, x{:}K, \Gamma_2 \overset{d}{\vdash} J$ *into its subderivation* $\Gamma_1 \overset{\mathbf{k}_{\Gamma_1}(d)}{\vdash} K\mathbf{kind}$*;*
3. $\Gamma \overset{d}{\vdash} (x{:}K_1)K_2$ *into its subderivation* $\Gamma, x{:}K_1 \overset{\mathbf{p}_-{}^\circ(d)}{\vdash} K_2$*;*
4. $\Gamma \overset{d}{\vdash} K_1 = K_2$ *into the derivation* $\mathbf{l}_=(d)$ *of* $\Gamma \vdash K_1\mathbf{kind}$ *and* $\mathbf{r}_=(d)$ *of* $\Gamma \vdash K_2\mathbf{kind}$*;*
5. $\Gamma \overset{d}{\vdash} k_1 = k_2{:}K$ *into the derivation* $l^\circ_=(d)$ *of* $\Gamma \vdash k_1{:}K$ *and* $r^\circ_=(d)$ *of* $\Gamma \vdash k_2{:}K$*;*
6. $\Gamma \overset{d}{\vdash} \Sigma{:}K$ *into the derivation* $\mathbf{kd}(d)$ *of* $\Gamma \vdash K\mathbf{kind}$ *(*Σ *denotes here term or term equality);*
7. $\Gamma \overset{d}{\vdash} A <_c B{:}\mathbf{Type}$ *or* $\Gamma \overset{d}{\vdash} K <_c K'$ *into the derivations* $\mathbf{l}_<(d)$ *of* $\Gamma \vdash A{:}\mathbf{Type}$ *or* $\Gamma \vdash K\mathbf{kind}$*,* $\mathbf{r}_<(d)$ *of* $\Gamma \vdash B{:}\mathbf{Type}$ *or* $\Gamma \vdash K'\mathbf{kind}$*,* $\mathbf{co}(d)$ *of* $\Gamma \vdash c{:}(El(A))El(B)$ *or* $\Gamma \vdash c{:}(K)K'$ *respectively.*

Remark 3.3 *The derivation* d' *obtained from* d *by any of the algorithms mentioned in the theorems 3.1 and 3.2 contains only the applications of the rules present in* d *(except the rules that are eliminated) and some new applications of the* LF^- *rules.*

Corollary 3.4 *Taking into account remark 3.3, the theorems above may be applied to any subsystem of* $T[\mathcal{R}]$ *that contains the* LF*-rules.*

3.2 Transitivity elimination

The role of the following theorem is to expand coherence conditions "half-way", from subtyping judgements in $T[\mathcal{R}]_0$ to the subkinding judgements in its conservative extension $T[\mathcal{R}]_{0K}$. (Combination of this result with coercion completion, which will transform arbitrary subkinding judgements into judgements that belong to $T[\mathcal{R}]_{0K}$ is then used in the proof of main results.)

Theorem 3.5 *(Elimination of transitivity of subkinding in* $T[\mathcal{R}]_{0K}$*.) There is an algorithm, transforming every derivation of the judgement* $\Gamma \vdash K <_c K'$ *in* $T[\mathcal{R}]_{0K}$ *into a derivation of the judgement* $\Gamma \vdash K <_{c'} K'$ *in the same calculus, not containing rules* $SK.5-7$*, and such that* $\Gamma \vdash c' = c{:}(K)K'$ *in* T*. Moreover, if congruence rule was not used then the derivation of* $c = c'$ *uses only* β*-equality rule 5.7.*

Corollary 3.6 *If the coherence conditions for subtyping are satisfied, then in* $T[\mathcal{R}]_{0K}$ *coherence also holds in the following sense:*

a) if $\Gamma \vdash K = K'$ *in* T *(or* $T[\mathcal{R}]_{0K}$*, since the latter is conservative extension of* T*) then* $\Gamma \vdash K <_c K'$ *is not derivable in* $T[\mathcal{R}]_{0K}$*;*

b) if $\Gamma \vdash K <_c K'$ *and* $\Gamma \vdash K <_{c'} K'$ *are derivable in* $T[\mathcal{R}]_{0K}$*, then* $\Gamma \vdash c = c' : (K)K'$

4 Coercion completion and conservativity

4.1 The problem of conservativity

We consider in this paper the conservativity problem, one of the main problems when an approach to coercive subtyping as an abbreviational mechanism is taken.

The issue of conservativity is closely connected with that of coherence of coercions (uniqueness of coercion between given subtype and supertype). Suppose $f{:}(x{:}K)K'$ and $a{:}K_0$, where $K_0 <_c K'$. By coercive definition rule, $f(c(a)) = f(a)$. If there is another coercion $K_0 <_{c'} K$ such that $c \neq c'$ (assume c, c' be the terms of T), then one may derive that $f(c(a)) = f(c'(a))$, and, with $f \equiv [x{:}K]x$ and $a \equiv y\ [y{:}K_0]$, $y{:}K_0 \vdash c(y) = c'(y){:}K$ and $\vdash c = c' : (K_0)K$. In this case, the extension would not be a conservative extension. It may even be the case that the whole system becomes inconsistent.

Conceptually, the basic coercions are declared by the user. This part is open for extensions. Thus, one may require coherence of the basic coercions (it is "user's responsibility")[3]. Still non-conservativity might arise if derived coercions were not coherent. The problem is how the conditions on basic coercions are connected with the behaviour of arbitrary coercions. In a dependent type system it is even more complicated by the fact that the coercions in question may be themselves derived using coercive rules.

The purpose of our work is to show how coherence conditions do imply conservativity.

Remark 4.1 *It is possible to distinguish coercive application from ordinary application syntactically. To make distinction one may insert a "placeholder" for absent coercion as the "dot" in the following alternative formulation of coercive rules.* [4]

$$(CA.1')\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k{:}K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f \cdot (k){:}[c(k)/x]K'}$$

$$(CA.2')\frac{\Gamma \vdash f = f'{:}(x{:}K)K' \quad \Gamma \vdash k_1 = k_2{:}K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f \cdot (k_1) = f' \cdot (k_2){:}[c(k_1)/x]K'}$$

[3] We note that research on coherence checking and coherence proofs is on-going and beyond the scope of this paper.

[4] There are arguments in favor (and against) both variants. The formulation without placeholder for coercions is closer to standard practice.

$$(CD')\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k_0{:}K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f \cdot (k_0) = f(c(k_0)){:}[c(k_0)/x]K'}$$

This remark is important for the conservativity problem. Since with coercive rules new expressions (containing coercive application) become derivable, the notion of conservativity for the system where coercive applications are not marked requires some subtlety.

The conservativity for the system with the 'dot'-marking can be understood in ordinary sense. However, the conservativity for the system without the 'dot'-marking should be understood as the statement "every judgement J which is derivable in the system with coercive rules and doesn't contain coercive applications is derivable in the system without coercive rules". *This formulation presupposes that it is possible to check if J contains coercive applications, at least when a derivation of J is given. (Note that the presence of coercive rules in a derivation d doesn't a-priori mean that its final judgement contains coercive applications.)*

As we show, in the situation considered in present paper (when basic coercions are coherent) the check for coercive applications is always possible and two formulations are equivalent. (The calculus with "dot" is equivalent to the calculus without dot in the sense that there is one-to-one correspondence between derivations.)

4.2 Coercion completion: the idea

Consider an inference of coercive application rule $CA.1$.

$$\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k{:}K_0 \quad \Gamma \vdash K_0 <_c K}{\Gamma \vdash f(k){:}[c(k)/x]K'}$$

Assume that there exist some T-derivations of the premises $\Gamma \vdash f{:}(x{:}K)K'$ and $\Gamma \vdash k{:}K_0$, and a T-derivation of the judgement $\Gamma \vdash c{:}(K_0)K$. Now we obtain a T-derivation as follows:

$$\frac{\dfrac{}{\Gamma \vdash f{:}(x{:}K)K'} \quad \dfrac{\Gamma \vdash k{:}K_0 \quad \Gamma \vdash c{:}(K_0)K}{\Gamma \vdash c(k){:}K}}{\Gamma \vdash f(c(k)){:}[c(k)/x]K'}.$$

Coercive equality application rule and coercive definition rule could be modified in the same way.

This construction suggests an idea how to define a transformation (we call it $\mathbf{\Theta}$) on the whole derivation. We should begin from the top, and move to the bottom replacing subkinding judgements in the premises of coercive rules CA.1, CA.2 and CD by the derivations of their coercion terms, and modifying the rules accordingly.

The problem with this idea is that the insertions of coercions depend on derivations. Thus, it should be shown that after insertion the premises of all rules will be matching (at least, up to the equality in T). Note, that even the

identical kinds or terms may be modified in different ways in different derivations, since different coercion terms could be inserted. This is one of the places where the coherence conditions are to be used.

For example, if we consider the ordinary application rule

$$\frac{\Gamma \vdash f{:}(x{:}K_1)K_2 \quad \Gamma \vdash k{:}K_1}{\Gamma \vdash f(k){:}[k/x]K_2},$$

and assume that some $T[\mathcal{R}]$-derivations of its premises, say, d_1, d_2 became the derivations $\mathbf{\Theta}(d_1), \mathbf{\Theta}(d_2)$ in T of $\Gamma' \vdash f'{:}(x{:}K_1')K_2'$ and $\Gamma'' \vdash k''{:}K_1''$ respectively, then the corresponding kinds in Γ' and Γ'' should be equal in T, and the same for K_1' and K_1''. If they are T-equal, there is a canonical way to insert a T-derivation of the corresponding equality (or equalities), and then use the same rule as in the main derivation.

If the transformation is defined, the result is a T-derivation for the judgements which are not subtyping or subkinding. If it is defined for a derivation of a subtyping or subkinding judgement, the result is a derivation in $T[\mathcal{R}]_0$ or $T[\mathcal{R}]_{0K}$ (conservative extensions of T that have corresponding judgement forms but do not use coercive rules CA.1, CA.2, CD).

4.3 Coercion completion and conservativity proof

Let us outline shortly the structure of conservativity proof based on coercion completion. The main technical results are presented in section 5.

- In the subsection 5.2 we define the operation $\mathbf{\Theta}$, used in coercion completion. The definition, in addition to the idea of inserting the coercion terms in place of coercive applications, uses the idea that after the insertion the premises of ordinary rules are to be "adjusted" (using T-equalities), in order to make possible the use of the same rule. At this stage $\mathbf{\Theta}$ is defined as a partial operation, since it is not yet proved that one may always derive in T the necessary equalities.

- In the subsection 5.3 we show that the operation $\mathbf{\Theta}$ agrees with the algorithms considered in the theorems 3.1 and 3.2. For example, if for some derivation d the derivation $\mathbf{\Theta}(d)$ is defined (it doesn't contain coercive rules), and δ is any of the algorithms considered in the theorems 3.1, 3.2, then $\mathbf{\Theta}(\delta(d))$ is also defined, and the final judgements of $\mathbf{\Theta}(d)$ and $\mathbf{\Theta}(\delta(d))$ are equal in T (componentwise).

- In the subsection 5.4 we prove that $\mathbf{\Theta}$ is total. Main idea: if there is a rule with several premises, their matching parts may be considered as presupposed judgements of the premises. If $\mathbf{\Theta}$ is defined for the premises, one may use the results of previous section to show that after the use of these operations the premises are still matching up to T-equality. Coherence of coercions in $T[\mathcal{R}]_{0K}$ (corollary 3.6) is also used here.

- Totality of $\mathbf{\Theta}$ permits us to prove that if a judgement J is derivable then the places where coercions are inserted do not depend on its derivation, if J is derivable in the system with coercions then $\mathbf{\Theta}(J)$ may be defined up to T-equality independently on derivation and finally to prove conservativity (in the sense discussed above, that is, if J is not modified by $\mathbf{\Theta}$, it is derivable in T).

4.4 The scope of conservativity proof

The technique we use in the proof of conservativity is very general, in particular, it doesn't depend on any assumptions about normalization properties and covers subtyping rules naturally connected with inductive types. For instance, the proof works for the subtyping schemata based on the schemata for inductive types [Luo99]. In the following, we illustrate the subtyping schemata with the example of Σ-types, and explain the difficulty to be encountered in the proof.

Subtyping schemata. There is a general schema to define the rules of $T[\mathcal{R}]_0$ that 'propagate' the coercions between parameter types to parameterised inductive types. If $\overline{\Xi}[A_1, ..., A_n]$ is a list of inductive schemata depending on type parameters $A_1, ..., A_n$ (each parameter occurs only covariantly or contravariantly) and for some types $B_1, ..., B_n, C_1, ..., C_n$ there are basic coercions $B_i <_{c_i} C_i$ (if the parameter A_i is covariant) or $C_i <_{c_i} B_i$ (if it is contravariant) then one may declare the coercion

$$\mathcal{M}[\overline{\Xi}[B_1, ..., B_n]] <_c \mathcal{M}[\overline{\Xi}[C_1, ..., C_n]]$$

where $\mathcal{M}[...]$ denotes the inductive type corresponding to the schema with the parameters $B_1, ..., B_n, C_1, ..., C_n$ respectively and c is constructed from $c_1, ..., c_n$ according to certain rules. The following is an example illustrating this schema.

Example 4.2 *(Coercion between parameterised Σ-types.) With the notation of inductive schemata (see [Luo94]), the Σ-types (types of dependent pairs) is represented as:*

$$\begin{array}{rcl} \Sigma & =_{df} & [A{:}\mathbf{Type}][B{:}(A)\mathbf{Type}]\mathcal{M}[(x{:}A)(B(x))X] \\ & : & (A{:}\mathbf{Type})(B{:}(A)\mathbf{Type})\mathbf{Type}. \end{array}$$

The constructor $pair_\Sigma$ and elimination operator E_Σ are:

$$\begin{array}{rcl} pair_\Sigma & : & (A{:}\mathbf{Type})(B{:}(A)\mathbf{Type})(x{:}A)(B(x))\Sigma(A,B) \\ E_\Sigma & : & (A{:}\mathbf{Type})(B{:}(A)\mathbf{Type})(C{:}\Sigma(A,B))\mathbf{Type}) \\ & & (f{:}(x{:}A)(y{:}B(x))C(pair_\Sigma(x,y))) \\ & & (z{:}\Sigma(A,B))C(z) \end{array}$$

According to the subtyping schemata, one of the rules that propagate the coercions $A' <_{c_1} A''$, $B'(x) <_{c_2} B''(x)$ to Σ-types is:

$$\frac{\Gamma \vdash A' <_{c_1} A''{:}\mathbf{Type} \quad \Gamma, x{:}A' \vdash B'(x) <_{c_2} B''(x){:}\mathbf{Type}(c_1(x))}{\Sigma(A', B') <_c \Sigma(A'', B'')}$$

where c is the following term (in context Γ):

$$E_\Sigma(A', B', [u{:}\Sigma(A', B')]\Sigma(A'', B''), [x{:}A'][y{:}B'(x)](pair_\Sigma(A'', B'')(c_1(x), c_2(y))))$$

that is, $c(pair_\Sigma(A', B', x, y)) = pair_\Sigma(A'', B'', c_1(x), c_2(y))$.

The problem of matching premises. As it was pointed out above, the insertion of coercion terms may change the premises of the rules (if we proceed downwards a derivation) in such a way that they may not be matching as needed. However, we have some means to *make* modified premises matching. Namely, if the modified premises are derivable in T (or in $T[\mathcal{R}]_0, T[\mathcal{R}]_{0K}$), we may further modify them in order to make them matching again by adding derivable T-equalities.

The possibility to express matching in terms of identity of certain presupposed judgements was essential for our work. For example, if we consider the rule

$$\frac{\Gamma \vdash f{:}(x{:}K_1)K_2 \quad \Gamma \vdash t{:}K_1}{\Gamma \vdash f(t){:}[t/x]K_2} \quad (5.5)$$

and the premises are modified to $\Gamma' \vdash f'{:}(x{:}K_1')K_2'$ and $\Gamma'' \vdash t'{:}K_1''$ during the process of insertion of coercions; the condition of applicability of the same rule to the modified premises is that $\Gamma' \equiv \Gamma''$, $K_1' \equiv K_1''$. In fact, this may be regarded as the condition imposed on the presupposed judgements $\Gamma' \vdash K_1'\mathbf{kind}$ and $\Gamma'' \vdash K_1''\mathbf{kind}$ of the judgements $\Gamma' \vdash f'{:}(x{:}K_1')K_2'$ and $\Gamma'' \vdash t'{:}K_1''$.

It is possible to express the conditions of applicability of the rules to given premises in terms of identity of certain presupposed judgements of the premises for specifications in type theories such as UTT or Martin-Löf intensional type theory.

The method of coercion completion we describe below works well with concrete forms of the rules we considered, such as the rules introducing constant coercions, coercions parameterised by elements of a type or the rules "propagating" coercions from the level of type parameters to the level of an inductive type.

However, it remains an open problem to find some simple (sufficient) general conditions on the form of rules of type theory T and of the system $\mathcal{R}$ of basic subtyping rules such that coercion completion will work.

5 Coercion Completion and its Properties

In this section we define the operation $\mathbf{\Theta}$ used in coercion completion and present the main results leading to the results about conservativity in the end of the paper.

5.1 Notational Conventions.

Let $\Gamma' = x_1{:}K'_1, x_2{:}K'_2, ..., x_n{:}K'_n$, $\Gamma'' = x_1{:}K''_1, x_2{:}K''_2, ..., x_n{:}K''_n$. We shall use the abbreviation $\Gamma \vdash \Gamma' = \Gamma''$, meaning

$$\Gamma \vdash K'_1 = K''_1,\ \Gamma, x_1{:}K''_1 \vdash K'_2 = K''_2,\ ...\ \Gamma, x_1{:}K''_1, ..., x_{n-1}{:}K''_{n-1} \vdash K'_n = K''_n.$$

We shall also consider equality between judgements of the same form (again as abbreviation). It will include the equality between contexts introduced above and the equality between corresponding parts of the right side. One should be careful in what order these equalities are considered. For example, if we write $\Gamma \vdash k{:}K = \Gamma' \vdash k'{:}K'$ it should be $\vdash \Gamma = \Gamma'$; if it is so, $\Gamma \vdash K = K'$ makes sense and it should be true. In its turn,$\Gamma \vdash k = k'{:}K$ should be true.

Definition 5.1 *Let d, d' be some derivations. Then $d \sim d'$ will mean that the final judgements of d, d' are equal in the sense described above.*

For inference rules, we shall use the abbreviation

$$\frac{\Gamma, \Gamma' \vdash J \quad \Gamma \vdash \Gamma' = \Gamma''}{\Gamma, \Gamma'' \vdash J}(3.3),$$

meaning the the following inference

$$\cfrac{\cfrac{\cfrac{\Gamma, x_1{:}K'_1, x_2{:}K'_2, ..., x_n{:}K'_n \vdash J \quad \Gamma \vdash K'_1 = K''_1}{\Gamma, x_1{:}K''_1, x_2{:}K'_2, ..., x_n{:}K'_n \vdash J}}{.....} \qquad \cfrac{}{\Gamma, x_1{:}K''_1, ..., x_{n-1}{:}K''_{n-1} \vdash K'_n = K''_n}}{\Gamma, x_1{:}K''_1, x_2{:}K''_2, ..., x_n{:}K''_n \vdash J}$$

(... means an appropriate series of replacement of kinds by their equal using "retyping" rule (3.3)).

Below, if not stated otherwise, equality means T-equality.

5.2 Definition of Θ.

Θ is the transformation (coercion completion) that inserts coercions into ordinary derivations (coercive rules disappear). The derivations may include substitutions, *wkn*, context-retyping 3.3 etc. In general, how a judgement is modified depends on its derivation. $\Theta_d(J)$ will denote the judgement J in d modified by Θ.

Note that we cannot require matching of the premises up to identity, since inserted coercion terms from different branches may be only equal (and not identical) even if the places where they should be inserted coincide (that also cannot be guaranteed in advance).

Θ is defined by structural induction, and the definition includes the assumption (routinely verified) that the *form* of the judgement is not changed.

- For all $T[\mathcal{R}]_{0K}$-derivations d, $\Theta(d) \equiv d$.

- If d ends by rule $\mathcal{R}$ with one premise, $d \equiv \dfrac{\dfrac{d_0}{J}}{R(J)}$, and $\mathbf{\Theta}(d_0)$ is defined, then $\mathbf{\Theta}(d) \equiv \dfrac{\mathbf{\Theta}(\dfrac{d_0}{J})}{R(\mathbf{\Theta}_{d_0}(J))}$. (One may check that $\mathcal{R}$ will be applicable, since the structure of J is not changed.) This case embraces the rules 1.2, 1.3, 2.1, 2.2, 2.4, 2.5, 4.1-4.3, 5.1, 5.3, 5.8, SK.1, declarations of constants in the type theory T etc.

- If $d \equiv \dfrac{\dfrac{d_1}{J_1} \quad \ldots \quad \dfrac{d_k}{J_k}}{R(J_1, \ldots, J_k)}$ where $\mathcal{R}$ is different from $CA.1, CA.2, CD$ and $\mathbf{\Theta}(d_i)$ is defined $(1 \leq i \leq k)$, then $\mathbf{\Theta}(d)$ is defined using the following scheme:

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{\Theta}(d_1)}{\mathbf{\Theta}_{d_1}(J_1)} \quad \ldots \quad \dfrac{\mathbf{\Theta}(d_k)}{\mathbf{\Theta}_{d_k}(J_k)} \quad \dfrac{?(\mathbf{T-derivations})}{\mathbf{Equalities}}}{=-\mathbf{transitivity\ and\ retyping\ rules}}}{J'_1 \ldots\ldots J'_k}}{R(J'_1 \ldots\ldots J'_k)}$$

  This schema should be explained and some details should be fixed. (Illustrations see below.) First of all, $J'_1, \ldots, J'_k$ are obtained in certain deterministic way from $\mathbf{\Theta}_{d_1}(J_1), \ldots, \mathbf{\Theta}_{d_k}(J_k)$. Determinism is achieved by using only the equalities between the constituents of the judgements $\mathbf{\Theta}_{d_1}(J_1), \ldots, \mathbf{\Theta}_{d_k}(J_k)$ and imposing certain order of preference (usually a kind is changed to an equal kind from some judgement that is to the left).

  The number of equalities to be considered is finite. Their derivations in T are to be found but the equalities themselves depend only on the judgements.

  The equalities are used to make possible application of the same rule $\mathcal{R}$ as in d. If necessary equalities are not derivable in T, then $\mathbf{\Theta}(d)$ is not defined. Since the equalities whose T-derivations should be found are determined by $\mathbf{\Theta}_{d_1}(J_1), \ldots, \mathbf{\Theta}_{d_k}(J_k)$, the choice of T-derivations does not affect the conclusion of $\mathcal{R}$.

- Let $d \equiv \dfrac{\Gamma \overset{d_1}{\vdash} f{:}(x{:}Q')K \quad \Gamma \overset{d_2}{\vdash} k{:}Q \quad \Gamma \overset{d_3}{\vdash} Q{<_c}Q'}{\Gamma \vdash f(k){:}[c(k)/x]K} CA.1$. In this case we shall write definition of $\mathbf{\Theta}$ explicitly. (It may serve also as illustration of how the equalities and retyping rules are used.)

  Let $\Gamma_1, \Gamma_2, \ldots$ etc denote $\mathbf{\Theta}_{d_1}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma), \ldots$ respectively. Applying $\mathbf{\Theta}$ to the derivations d_1, d_2, d_3, we obtain the derivations $\Gamma_1 \overset{\mathbf{\Theta}(d_1)}{\vdash} f_1{:}((x{:}Q'_1)K_1)$,

$\Gamma_2 \overset{\Theta(d_2)}{\vdash} k_2{:}Q_2$ and $\Gamma_3 \overset{\Theta(d_3)}{\vdash} Q <_{c_3} Q'_3$. Let h be

$$\dfrac{\dfrac{\dfrac{\dfrac{\mathbf{co}(\Theta(d_3))}{\Gamma_3 \vdash c_3{:}(Q_3)Q'_3} \quad \dfrac{?_1}{\vdash \Gamma_3 = \Gamma_1}}{\Gamma_1 \vdash c_3{:}(Q_3)Q'_3} \quad \dfrac{\dfrac{\dfrac{\Theta(d_2)}{\Gamma_2 \vdash k_2{:}Q_2} \quad \dfrac{?_2}{\vdash \Gamma_2 = \Gamma_1}}{\Gamma_1 \vdash k_2{:}Q_2} \quad \dfrac{?_3}{\Gamma_1 \vdash Q_2 = Q_3}}{\Gamma_1 \vdash k_2{:}Q_3}}{\Gamma_1 \vdash c_3(k_2){:}Q'_3} \quad \dfrac{?_4}{\Gamma_1 \vdash Q'_3 = Q'_1}}{\Gamma_1 \vdash c_3(k_2){:}Q'_1}$$

As $\Theta(d)$ we take

$$\dfrac{\Gamma_1 \overset{\Theta(d_1)}{\vdash} f_1{:}(x{:}Q'_1)K_1 \quad \Gamma_1 \overset{h}{\vdash} c_3(k_2){:}Q'_1}{\Gamma_1 \vdash f_1(c_3(k_2)){:}[c_3(k_2)/x]K_1}$$

(Note the use of the algorithm **co** applied to T-derivation $\Theta(d_3)$. The question-marks correspond to T-derivations that should be found if Θ is defined.)

- No principal differences in cases CA.2 and CD.

Lemma 5.2 *If $\Theta(d)$ is defined it is defined for any subderivation d' of d.*

5.3 Coercion Completion and Transformations of Derivations

Let us call constituents of a judgement $\Gamma \vdash J$

- the kinds of variables from Γ;
- the kinds of bound variables (i.e., for every occurrence of the binder $[x{:}K]$ or $(x{:}K)$ the kind K is a constituent);
- if $J \equiv K\mathbf{kind}$ or $J \equiv K_1 = K_2$ then the kind K, respectively K_1, K_2;
- if $J \equiv k{:}K$ or $J \equiv k_1 = k_2{:}K$, then k, K, respectively k_1, k_2, K;
- if $J \equiv A <_c B{:}\mathbf{Type}$ then $c, A, B, \mathbf{Type}$;
- if $J \equiv K_1 <_c K_2$ then c, K_1, K_2;
- any subkinds of above mentioned kinds.

The A, B, c, k, k_1, k_2 will be called term-constituents and the rest kind-constituents of $\Gamma \vdash J$. These constituents will be considered as different in different occurrences (e.g., labelled by occurrences). Note that we do not consider proper subterms as constituents.

In this paper the presupposed judgements are the principal example of the judgements built of constituents of some main judgement.

When some derivation d of a judgement $\Gamma \vdash J$ is given, and $\Theta(d)$ is defined, it will determine also unique syntactic transformation of all constituents of $\Gamma \vdash J$.

Let $\mathbf{\Theta}_d(\Gamma \vdash J)$ be the final judgement of $\mathbf{\Theta}(d)$. Let $\Gamma' \vdash J'$ be any judgement (not necessarily derivable) where Γ' is obtained by kinding of free variables by kind-constituents of Γ, and J' one of possible forms of judgements built of kind and/or term-constituents of $\Gamma \vdash J$. We shall denote by $\mathbf{\Theta}_d(\Gamma' \vdash J')$ the result of replacement of each constituent of $\Gamma \vdash J$ which was used in $\Gamma' \vdash J'$ by the corresponding part of $\mathbf{\Theta}_d(\Gamma \vdash J)$.

Lemma 5.3 *Let d be a derivation.*

a) If $\mathbf{\Theta}(d)$ is defined it ends by the same rule as d.

b) If d_0 is a subderivation of d, then $\mathbf{\Theta}(d_0)$ is defined.

c) If d'_0 is some derivation of the same judgement as d_0 (say, J), d' is obtained from d by replacement of d_0 by d'_0, $\mathbf{\Theta}(d'_0)$ is defined and $\mathbf{\Theta}(d_0) \sim \mathbf{\Theta}(d'_0)$, then $\mathbf{\Theta}(d')$ is defined and $\mathbf{\Theta}(d) \sim \mathbf{\Theta}(d')$.

Lemma 5.4 *Let d be a derivation of $\Gamma, \Gamma' \vdash J$, $\Gamma, x{:}K, \Gamma' \vdash J$ or $\Gamma \vdash (x{:}K_1)K_2$ in $T[\mathcal{R}]^-$. If $\mathbf{\Theta}(d)$ is defined then $\mathbf{\Theta}(\mathbf{v}_\Gamma(d))$, $\mathbf{\Theta}(\mathbf{k}_{\Gamma'}(d))$, $\mathbf{\Theta}(\mathbf{p}\_{}^\circ\Gamma(d))$ are defined and:*

- $\mathbf{\Theta}_{\mathbf{v}_\Gamma(d)}(\Gamma \vdash \mathbf{valid}) =_T \mathbf{\Theta}_d(\Gamma \vdash \mathbf{valid})$
- $\mathbf{\Theta}_{\mathbf{k}_\Gamma(d)}(\Gamma \vdash K\mathbf{kind}) =_T \mathbf{\Theta}_d(\Gamma \vdash K\mathbf{kind})$,
- $\mathbf{\Theta}_{\mathbf{p}\_^\circ(d)}(\Gamma, x{:}K_1 \vdash K_2\mathbf{kind}) =_T \mathbf{\Theta}_d(\Gamma, x{:}K_1 \vdash K_2\mathbf{kind})$.

Theorem 5.5 *Let d be a $T[\mathcal{R}]$-derivation, and $\mathbf{E}$ be the elimination algorithm from the theorem 3.1. If $\mathbf{\Theta}(d)$ is defined, then $\mathbf{\Theta}(\mathbf{E}(d))$ is, and $\mathbf{\Theta}(\mathbf{E}(d)) \sim \mathbf{\Theta}(d)$.*

Proof. The proof follows the structure of the proof of the theorem 3.1 (see [JLS98]), where first the elimination of one wkn, one context-retyping etc was considered. Let us consider as illustration one-rule elimination. There are standard and special cases.

In standard case d has the form $\dfrac{d_1 \quad ...d_k}{J}r$, where $d_1, ..., d_k$ are the derivations of the premises and the derivation d_1 itself may be written as $\dfrac{\dfrac{D_1}{J_1} \quad ...\dfrac{D_m}{J_m}}{J'}R$. Here r is the rule to be eliminated and $\mathcal{R}$ is the last rule of d_1. The derivations $d_1, ..., d_k$ belong to $T[\mathcal{R}]^-$.

In such a case one step of elimination is defined by

$$\mathbf{E}(d) \equiv \mathbf{E}(\frac{\dfrac{D_1 \quad ...D_m}{J_1}R \quad d_2 \quad ...d_k}{J}) \equiv \frac{\mathbf{E}(\dfrac{D_1 \quad d_2...d_k}{J_1}r) \quad ...\mathbf{E}(\dfrac{D_m \quad d_2...d_k}{J_m}r)}{J}R$$

This elimination step is divided naturally into two parts. Schematically,

$$d \mapsto d' \equiv \frac{\dfrac{D_1 \quad d_2...d_k}{J_1}r \quad ...\dfrac{D_m \quad d_2...d_k}{J_m}r}{J} \mapsto \frac{\mathbf{E}(\dfrac{D_1 \quad d_2...d_k}{J_1}r) \quad ...\mathbf{E}(\dfrac{D_m \quad d_2...d_k}{J_m}r)}{J}R$$

First, we check that if $\mathbf{\Theta}(d)$ is defined then $\mathbf{\Theta}(d')$ is, and the final judgements of $\mathbf{\Theta}(d)$ and $\mathbf{\Theta}(d')$ are equal [5]. Second, we apply I.H. to the premises of $\mathcal{R}$ in d' and use lemma 5.3.

Special cases are those where the end of d is restructured in different way. Let us consider one standard and one special case as illustrations.

One wkn-elimination. $d \equiv \dfrac{\Gamma, \Gamma' \overset{d_1}{\vdash} J \quad \Gamma, \Gamma'' \overset{d_2}{\vdash} \mathbf{valid}}{\Gamma, \Gamma'', \Gamma' \vdash J}(wkn)$, d_1, d_2 do not contain wkn.

Assume that $\mathbf{\Theta}(d)$ is defined. By definition of $\mathbf{\Theta}$, $\mathbf{\Theta}_d(\Gamma, \Gamma'', \Gamma' \vdash J) \equiv \mathbf{\Theta}_{d_1}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma''), \mathbf{\Theta}_{d_1}(\Gamma') \vdash \mathbf{\Theta}_{d_1}(J)$.

In general, $d_1 \equiv \dfrac{\Gamma, \Gamma'_1 \overset{e_1}{\vdash} J_1 \quad ... \Gamma, \Gamma'_k \overset{e_k}{\vdash} J_k}{\Gamma, \Gamma' \vdash J}(r)$.

The modified d is

$$d' \equiv \frac{\dfrac{\Gamma, \Gamma'_1 \overset{e_1}{\vdash} J_1 \quad \Gamma, \Gamma'' \overset{d_2}{\vdash} \mathbf{valid}}{\Gamma, \Gamma'', \Gamma'_1 \vdash J_1}(wkn) \ldots \dfrac{\Gamma, \Gamma'_k \overset{e_k}{\vdash} J_k \quad \Gamma, \Gamma'' \overset{d_2}{\vdash} \mathbf{valid}}{\Gamma, \Gamma'', \Gamma'_k \vdash J_k}(wkn)}{\Gamma, \Gamma'', \Gamma' \vdash J}(r).$$

Since $\mathbf{\Theta}(d_1)$ is defined, $(+)$ $\mathbf{\Theta}_{d_2}(\Gamma) = \mathbf{\Theta}_{e_1}(\Gamma) = \ldots \mathbf{\Theta}_{e_k}(\Gamma)$. Thus, $\mathbf{\Theta}(e'_i)$ is defined $(1 \leq i \leq k)$, where $e'_i \equiv \dfrac{\Gamma, \Gamma'_i \overset{e_i}{\vdash} J_i \quad \Gamma, \Gamma'' \overset{d_2}{\vdash} \mathbf{valid}}{\Gamma, \Gamma'', \Gamma'_i \vdash J_i}(wkn)$. $\mathbf{\Theta}(e'_i)$ is

$$\frac{\dfrac{\mathbf{\Theta}(e_i)}{\mathbf{\Theta}_{e_i}(\Gamma), \mathbf{\Theta}_{e_i}(\Gamma'_i) \vdash \mathbf{\Theta}_{e_i}(J)} \quad \dfrac{\mathbf{\Theta}_{d_2}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma'') \overset{\mathbf{\Theta}(d_2)}{\vdash} \mathbf{valid} \quad \overset{h_i}{\vdash} \mathbf{\Theta}_{d_2}(\Gamma) = \mathbf{\Theta}_{e_i}(\Gamma)}{\mathbf{\Theta}_{e_i}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma'') \vdash \mathbf{valid}}}{\mathbf{\Theta}_{e_i}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma''), \mathbf{\Theta}_{e_i}(\Gamma'_i) \vdash \mathbf{\Theta}_{e_i}(J)}.$$

Here h_i denote appropriate T-derivations obtained from the T-derivation of equalities $(+)$.

Before r may be applied to the conclusions of $\mathbf{\Theta}(e'_i)$, some T-equalities must be incerted. They are obtained by modification of the T-equalities used in $\mathbf{\Theta}(d_1)$.

Those T-equalities are (a) the equality judgements used to change kinds of the variables in $\mathbf{\Theta}_{e_i}(\Gamma)$ (their context is part of $\mathbf{\Theta}_{e_1}(\Gamma)$) and (b) the equalities of the form $\mathbf{\Theta}_{e_1}(\Gamma), \ldots \vdash \mathbf{\Theta}_{e_i}(K) = \mathbf{\Theta}_{e_1}(K')$. The equalities (a) may be used without modification. To the equalities of (b) wkn must be applied first:

$$\frac{\dfrac{}{\mathbf{\Theta}_{e_1}(\Gamma), \ldots \vdash \mathbf{\Theta}_{e_i}(K) = \mathbf{\Theta}_{e_1}(K')} \quad \dfrac{\mathbf{\Theta}_{d_2}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma'') \vdash \mathbf{valid} \quad \vdash \mathbf{\Theta}_{d_2}(\Gamma) = \mathbf{\Theta}_{e_1}(\Gamma)}{\mathbf{\Theta}_{e_1}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma'') \vdash \mathbf{valid}}}{\mathbf{\Theta}_{e_1}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma''), \ldots \vdash \mathbf{\Theta}_{e_i}(K')}$$

and then they may be used as in $\mathbf{\Theta}(d_1)$

Thus, $\mathbf{\Theta}(d')$ is defined. By I.H. $\mathbf{\Theta}(e'_i) \sim \mathbf{\Theta}(\mathbf{E}(e'_i))$, and we use lemma 5.3, c).

[5] In most, but not all cases they are identical

The only exceptional case for wkn is $d_1 \equiv \dfrac{\Gamma_0 \overset{d_0}{\vdash} K\mathbf{kind}}{\Gamma_0, x{:}K \vdash \mathbf{valid}}(1.2), \Gamma \equiv \Gamma_0, x{:}K$. In this case the derivation d_2 is taken as $\mathbf{E}(d)$. $\mathbf{\Theta}(\mathbf{E}(d))$ is defined because d_2 is subderivation of d, and $\mathbf{\Theta}(d)$ is defined (lemma 5.4). By the same lemma, $\mathbf{\Theta}_{d_2}(\Gamma) = \mathbf{\Theta}_d(\Gamma)$ and $\mathbf{\Theta}_{d_2}(\Gamma, \Gamma'') \equiv \mathbf{\Theta}_{d_2}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma'') = \mathbf{\Theta}_{d_1}(\Gamma), \mathbf{\Theta}_{d_2}(\Gamma'') \equiv \mathbf{\Theta}_d(\Gamma, \Gamma'')$.

It is one of the cases where equality (and not identity) of the conclusions of original and modified derivations arises. □

Theorem 5.6 *Let d be a $T[\mathcal{R}]$-derivation of one of the judgements:* 1. $\Gamma \vdash K_1 = K_2$ *;* 2. $\Gamma \vdash k_1 = k_2{:}K$*;* 3. $\Gamma \vdash \Sigma{:}K$ *(Σ denotes a term or term equality).;* 4. $\Gamma \overset{d}{\vdash} A <_c B{:}\mathbf{Type}$ *or* $\Gamma \overset{d}{\vdash} K <_c K'$

If $\mathbf{\Theta}(d)$ is defined then $\mathbf{\Theta}(\mathbf{l}_=(d)), \mathbf{\Theta}(\mathbf{r}_=(d)), \mathbf{\Theta}(l^\circ_=(d)), \mathbf{\Theta}(r^\circ_=(d)), \mathbf{\Theta}(\mathbf{kd}(d))$ are defined and

1. $\mathbf{\Theta}_d(\Gamma \vdash K_1\mathbf{kind}) =_T \mathbf{\Theta}_{\mathbf{l}_=(d)}(\Gamma \vdash K_1\mathbf{kind})$,
   $\mathbf{\Theta}_d(\Gamma \vdash K_2\mathbf{kind}) =_T \mathbf{\Theta}_{\mathbf{r}_=(d)}(\Gamma \vdash K_2\mathbf{kind})$;
2. $\mathbf{\Theta}_d(\Gamma \vdash k_1{:}K) =_T \mathbf{\Theta}_{l^\circ_=(d)}(\Gamma \vdash k_1{:}K)$,
   $\mathbf{\Theta}_d(\Gamma \vdash k_1{:}K) =_T \mathbf{\Theta}_{r^\circ_=(d)}(\Gamma \vdash k_1{:}K)$;
3. $\mathbf{\Theta}_d(\Gamma \vdash K\mathbf{kind}) =_T \mathbf{\Theta}_{\mathbf{kd}(d)}(\Gamma \vdash K\mathbf{kind})$;
4. $\mathbf{\Theta}_d(\Gamma \vdash A{:}\mathbf{Type}) =_T \mathbf{\Theta}_{\mathbf{l}_<(d)}(\Gamma \vdash A{:}\mathbf{Type})$,
   $\mathbf{\Theta}_d(\Gamma \vdash K\mathbf{kind}) =_T \mathbf{\Theta}_{\mathbf{l}_<(d)}(\Gamma \vdash K\mathbf{kind})$,
   $\mathbf{\Theta}_d(\Gamma \vdash B{:}\mathbf{Type}) =_T \mathbf{\Theta}_{\mathbf{r}_<(d)}(\Gamma \vdash B{:}\mathbf{Type})$,
   $\mathbf{\Theta}_d(\Gamma \vdash K\mathbf{kind}) =_T \mathbf{\Theta}_{\mathbf{r}_<(d)}(\Gamma \vdash K\mathbf{kind})$,
   $\mathbf{\Theta}_d(\Gamma \vdash c{:}(El(A))El(B)) =_T \mathbf{\Theta}_{\mathbf{co}(d)}(\Gamma \vdash c{:}(El(A))El(B))$,
   $\mathbf{\Theta}_d(\Gamma \vdash c{:}(K)K') =_T \mathbf{\Theta}_{\mathbf{co}(d)}(\Gamma \vdash c{:}(K)K')$, *respectively.*

Proof. Structural induction, that follows the cases in the proof of the theorem 3.2. As in theorem 5.5, one step of extraction of a proof of presupposed judgement can be split (roughly) in two parts: restructuring of the end of d (including use of the extraction algorithm on the subderivations) and use of elimination algorithm to get rid of wkn, substitutions and other rules that may appear. With respect to $\mathbf{\Theta}$, we should check that this preserves definedness of $\mathbf{\Theta}$ and the structure of the parts of final judgement (up to T-equality). Lemma 5.3 and theorem 5.5 (in connection with the elimination part) are used. Consider as illustration one rule:

$$d \equiv \frac{\Gamma, x{:}K_1 \overset{d_1}{\vdash} K_1' = K_2' \quad \Gamma \overset{d_2}{\vdash} K_1 = K_2}{\Gamma \vdash (x{:}K_1)K_1' = (x{:}K_2)K_2'}(5.2),$$

Consider more complex case of the derivation $\mathbf{r}_{=}(d)$ (the construction of $\mathbf{r}_{=}(d)$ is taken from [JLS98]):

$$d \mapsto d' \equiv \dfrac{\dfrac{\Gamma, x{:}K_1 \overset{\mathbf{r}_{=}(d_1)}{\vdash} K_2' \quad \Gamma \overset{d_2}{\vdash} K_1 = K_2}{\Gamma, x{:}K_2 \vdash K_2'}(3.3)}{\Gamma \vdash (x{:}K_2)K_2'}(5.1) \mapsto$$

$$\mapsto \mathbf{r}_{=}(d) \equiv \dfrac{\mathbf{E}(\dfrac{\Gamma, x{:}K_1 \overset{\mathbf{r}^{\circ}_{=}(d_1)}{\vdash} K_2' \quad \Gamma \overset{d_2}{\vdash} K_1 = K_2}{\Gamma, x{:}K_2 \vdash K_2'}(3.3))}{\Gamma \vdash (x{:}K_2)K_2'}(5.1)$$

We check, that if $\mathbf{\Theta}(d)$ is defined then $\mathbf{\Theta}(d')$ is defined and $\mathbf{\Theta}(d) \sim_T \mathbf{\Theta}(d')$ (using definition of $\mathbf{\Theta}$, I.H. and lemma 5.3). Then we show the same for $\mathbf{\Theta}(d')$ and $\mathbf{\Theta}(\mathbf{r}_{=}(d))$ using theorem 5.5 and again lemma 5.3. □

5.4 Θ is total

Lemma 5.7 *Assume that the coherence conditions for subtyping are satisfied. Let d, d' be the derivations in $T[\mathcal{R}]^-$ of the same judgement: a)$\Gamma \vdash$ **valid**; b)$\Gamma \vdash K$**kind**; c)or of two judgements $\Gamma \vdash k{:}K$ and $\Gamma \vdash k{:}K'$. If $\mathbf{\Theta}(d)$ and $\mathbf{\Theta}(d')$ are defined, then: a)$\mathbf{\Theta}_d(\Gamma \vdash$ **valid**$) =_T \mathbf{\Theta}_{d'}(\Gamma \vdash$ **valid**$)$; b)$\mathbf{\Theta}_d(\Gamma \vdash K$**kind**$) =_T \mathbf{\Theta}_{d'}(\Gamma \vdash K$**kind**$)$; c)$\mathbf{\Theta}_d(\Gamma \vdash k{:}K) =_T \mathbf{\Theta}_{d'}(\Gamma \vdash k{:}K')$.*

Proof. Induction on the sum of sizes of two derivations. Base is covered by the case when d, d' are T-derivations. If d, d' end by the same rule with the same premises, the inductive step presents no difficulty. Of the remaining cases the principal cases are of two coercive applications with different coercion terms and of coercive application in one derivation and ordinary application in another. It is of crucial importance that the second case is impossible. We shall take it as illustration. Assume that

$$d \equiv \dfrac{\Gamma \overset{d_1}{\vdash} f{:}(x{:}Q)K \quad \Gamma \overset{d_2}{\vdash} k{:}Q}{\Gamma \vdash f(k){:}[k/x]K}5.5$$

$$d' \equiv \dfrac{\Gamma \overset{d'_1}{\vdash} f{:}(x{:}Q')K' \quad \Gamma \overset{d'_2}{\vdash} k{:}Q'_0 \quad \Gamma \overset{d'_3}{\vdash} Q'_0 <_c Q'}{\Gamma \vdash f(k){:}[c(k)/x]K'}CA.1.$$

In the following equalities and subtyping relations we omit the contexts. Using I.H. and the fact that $\mathbf{\Theta}(d), \mathbf{\Theta}(d')$ are defined we obtain

(i)$\mathbf{\Theta}_{d_1}((x{:}Q)K) =_T \mathbf{\Theta}_{d'_1}((x{:}Q')K')$ which implies $\mathbf{\Theta}_{d_1}(Q) =_T \mathbf{\Theta}_{d'_1}(Q')$;

(ii)$\mathbf{\Theta}_{d_2}(Q) =_T \mathbf{\Theta}_{d'_2}(Q'_0)$;

(iii) $\mathbf{\Theta}_{d'_3}(Q'_0) <_{\mathbf{\Theta}_{d'_3}(c)} \mathbf{\Theta}_{d'_3}(Q')$ in $T[\mathcal{R}]^-_{0K}$.

(iv) Since $\mathbf{\Theta}$ is defined for d, d', we have also $\mathbf{\Theta}_{d'_2}(Q'_0) =_T \mathbf{\Theta}_{d'_3}(Q'_0)$ and $\mathbf{\Theta}_{d_1}(Q') =_T \mathbf{\Theta}_{d_3}(Q')$.

This means the possibility to have at the same time equality and subtyping relation for two kinds in $T[\mathcal{R}]^{-}_{0K}$, which contradicts corollary 3.6.

In case of d and d' ending by CA.1 with different coercion terms c, c' in third premises, using similar analysis we show that $\mathbf{\Theta}_d(c) = \mathbf{\Theta}_{d'}(c')$ in $T[\mathcal{R}]_{0K}$ and this provides T-equality of the conclusions. □

Theorem 5.8 *If $\mathcal{R}$ is coherent, then the transformations $\mathbf{\Theta}$ is defined for all derivations in $T[\mathcal{R}]$ and if d, d' are derivations of the same judgement $\Gamma \vdash J$, then $\mathbf{\Theta}(d) \sim \mathbf{\Theta}(d')$. Moreover, if d is a derivation in $T[\mathcal{R}]_{0K}$ (this includes the derivations in $T[\mathcal{R}]_0$ and T), its final judgement is not changed at all.*

Proof. The proof that $\mathbf{\Theta}$ is total goes by induction on the structure of derivations. Main idea is to combine lemma 5.7 with theorem 5.6.The proof of the inductive step (that is, the proof that if the derivation d ends by some rule r with $d_1, .., d_k$ being the subderivations of the premises then the assumption that $\mathbf{\Theta}(d_1), ..., \mathbf{\Theta}(d_k)$ is defined implies $\mathbf{\Theta}(d)$ is defined) may be reduced to the question of T-equality between certain presupposed judgements of the premises modified by $\mathbf{\Theta}$. All presupposed judgements have one of the forms considered in lemma 5.7. In the proof of the last statement of the theorem remark 3.3 has to be taken into account. □

6 Coercion completion and conservativity

Theorem 6.1 *Let J be a judgement derivable in $T[\mathcal{R}]$. If $\mathcal{R}$ is coherent, and J is not subtyping or subkinding judgement, it is derivable in T iff there exists d in $T[\mathcal{R}]$ such that $\mathbf{\Theta}_d(J) \equiv J$ (by theorem 5.8 it implies $\mathbf{\Theta}_f(J) \equiv J$ for every derivation f). The last condition for the judgements of subtyping or subkinding form is equivalent to their derivability in $T[\mathcal{R}]_0$ (respectively, $T[\mathcal{R}]_{0K}$).*

Proof. Let d be some derivation of J in $T[\mathcal{R}]$. Implication from left to right is trivial since the derivations that do not contain coercive rules are not changed. Suppose now that there exists d such that $\mathbf{\Theta}_d(J) \equiv J$. It is enough to observe that $\mathbf{\Theta}(d)$ is a derivation in T (respectively in $T[\mathcal{R}]_0, T[\mathcal{R}]_{0K}$ if the judgement has the form of subtyping or subkinding). □

Remark 6.2 *This theorem solves obviously the problem of conservativity as it was discussed in the subsection 4.1. Indeed, if J is derivable we have to take any its derivation in $T[\mathcal{R}]$ and apply the algorithm $\mathbf{\Theta}$. Then J does not contain coercive applications iff it is not changed. In this case it is derivable without coercive rules.*

Final remarks

To conclude it would be convenient to add some notes about the formulation of the type theory with coercions that uses "placeholder" (the system with "dot" from the remark 4.1).

In addition to the problem of conservativity for this system its relationship to the system without "dot" should be studied. In fact, it is far from obvious that the two systems are equivalent. One may trivially transform a derivation in the system with "dot" into derivation without "dot" (just erasing the "dots"). But it is not obvious that one may transform a derivation without "dots" into a derivation with "dot". Similarly to the insertion of coercions described above (the transformation $\mathbf{\Theta}$) it is to be proved that the premises of the rules will be still matching if we insert "dots" marking coercive applications.

Let us denote by θ the transformation that replaces ordinary coercive rules by their version with "dot" (moving downwards the derivation). E.g.,

$$(CA.1)\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k{:}K_0 \quad \Gamma \vdash K_0{<_c}K}{\Gamma \vdash f(k){:}[c(k)/x]K'}$$

is replaced by

$$(CA.1)\frac{\Gamma \vdash f{:}(x{:}K)K' \quad \Gamma \vdash k{:}K_0 \quad \Gamma \vdash K_0{<_c}K}{\Gamma \vdash f \cdot (k){:}[c(k)/x]K'}$$

etc. The transformation $\mathbf{\Theta^\bullet}$ would insert coercions in the derivation where the places to insert coercions are marked by "dots".

If it is proved that θ is total, then it is easy to prove that the calculus where coercive applications are distinguished syntactically from ordinary applications is equivalent to the calculus where they are not. In fact, the totality of θ can be proved, but it turns out that the proof uses the whole apparatus that we have already developed for $\mathbf{\Theta}$. It is based on the following (easy) lemma.

Lemma 6.3 *If $\mathbf{\Theta}(d)$ and $\theta(d)$ are defined, then $\mathbf{\Theta^\bullet}(\theta(d))$ is defined and the final judgements of $\mathbf{\Theta}(d)$ and $\mathbf{\Theta^\bullet}(\theta(d))$ coincide. If J is the final judgement of d, then $\mathbf{\Theta}_d(J)$ is the result of replacement of "dots" in $\theta_d(J)$ by coercion terms.*

This lemma "binds" the problem of totality of θ to the problem of totality of $\mathbf{\Theta}$ and, in our view, justifies the priority we have given to the consideration of $\mathbf{\Theta}$.

The proof of totality of $\mathbf{\Theta}$ provides as its corollary the proof of totality of θ and equivalence (one-to-one correspondence between derivations) of the systems with and without "dot".

The notation with "dot" marking coercive applications makes it possible to formulate conservativity problem in traditional way: *if the judgement J does not contain "dot" then it is derivable in the system with coercive rules CA.1', CA.2', CD' iff it is derivable in the system without these rules.*

Acknowledgements. The authors would like to thank Dr. Healfdene Goguen, Dr. James McKinna and Prof. Giuseppe Longo for helpful discussions.

References

[Bai96] A. Bailey. Lego with implicit coercions. 1996. Draft.

[Bai98] A. Bailey. *The Machine-checked Literate Formalisation of Algebra in Type Theory*. PhD thesis, University of Manchester, 1998.

[BCGS91] V. Breazu-Tannen, T. Coquand, C. Gunter, and A. Scedrov. Inheritance and explicit coercion. *Information and Computation*, 93, 1991.

[Cal99] P.C. Callaghan. Plastic: an implementation of typed LF with coercions. Talk given in the Annual Conf of TYPES'99, June 1999.

[Che96] Gang Chen. Dependent type system with subtyping. Technical report, Technical Report LIENS-96-27, Ecole Normale Superieure-Paris, 1996.

[Che98] Gang Chen. *Sous-typage, Conversion de Types et Elimination de la Transitivite*. PhD thesis, Université Paris VII, 1998.

[CL00] P. C. Callaghan and Z. Luo. Plastic: an implementation of typed LF with coercive subtyping and universes. Draft paper submitted, 2000.

[Coq96] Coq. *The Coq Proof Assistant Reference Manual (version 6.1)*. INRIA-Rocquencourt and CNRS-ENS Lyon, 1996.

[CPM90] Th. Coquand and Ch. Paulin-Mohring. Inductively defined types. *Lecture Notes in Computer Science*, 1990.

[Dyb91] P. Dybjer. Inductive sets and families in Martin-Löf's type theory and their set-theoretic semantics. In G. Huet and G. Plotkin, editors, *Logical Frameworks*. Cambridge University Press, 1991.

[JLS98] A. Jones, Z. Luo, and S. Soloviev. Some proof-theoretic and algorithmic aspects of coercive subtyping. *Types for proofs and programs (eds, E. Gimenez and C. Paulin-Mohring), Proc. of the Inter. Conf. TYPES'96, LNCS 1512*, 1998.

[LMS95] G. Longo, K. Milsted, and S. Soloviev. A logic of subtyping. In *Proc. of LICS'95*, 1995.

[LP92] Z. Luo and R. Pollack. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, Department of Computer Science, University of Edinburgh, 1992.

[LS99] Z. Luo and S. Soloviev. Dependent coercions. *The 8th Inter. Conf. on Category Theory and Computer Science(CTCS'99), Edinburgh, Scotland. Electronic Notes in Theoretical Computer Science*, 1999.

[Luo94] Z. Luo. *Computation and Reasoning: A Type Theory for Computer Science*. Oxford University Press, 1994.

[Luo97] Z. Luo. Coercive subtyping in type theory. *Proc. of CSL'96, the 1996 Annual Conference of the European Association for Computer Science Logic, Utrecht. LNCS 1258*, 1997.

[Luo99] Z. Luo. Coercive subtyping. *Journal of Logic and Computation*, 9(1):105–130, 1999.

[MCMS94] J.C. Mitchell, L. Cardelli, S. Martini, and A. Schedrov. An extension of system f with subtyping. *Information and Computation*, 1994.

[NPS90] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin-Löf's Type Theory: An Introduction.* Oxford University Press, 1990.

[PS94] B. Pierce and M. Steffen. Higher-order subtyping. *Proc. of IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET)*, 1994.

[Sai97] A. Saibi. Typing algorithm in type theory with inheritance. *Proc of POPL'97*, 1997.

[Tiu95] J. Tiurin. Equational axiomatization of bicoercibility for polymorphic types. *LNCS*, 1026, 1995.

Appendix

The following figure contains the rules of the typed logical framework LF with minor modifications. The main modification is that we included (to simplify some formulations) the rules of weakening wkn and context-retyping, which may be eliminated.

For the reference purposes the rules are given numbers. The rules may be divided into several groups: 1.1-1.3 and the weakening wkn are related to contexts and assumptions; 2.1-2.6 are general equality rules; 3.1 - 3.3 equality typing rules; 4.1 - 4.3 describe the kind **Type**; 5.1 - 5.8 are related to dependent product kinds; and 6.1 - 6.7 are substitutions.

Note that in the expressions like $[x{:}K]k$ and $(x{:}K)K'$ $[x{:}K]$ and $(x{:}K)$ play the role of binders (bind all free x in k, respectively K'). This notation is commonly used in the literature on proof assistants.

$$(1.1)\ \frac{}{<>\vdash \mathbf{valid}} \qquad (1.2)\ \frac{\Gamma\vdash K\mathbf{kind}\ \ x\notin FV(\Gamma)}{\Gamma,x{:}\,K\vdash \mathbf{valid}} \qquad (1.3)\frac{\Gamma,x{:}\,K,\Gamma'\vdash \mathbf{valid}}{\Gamma,x{:}\,K,\Gamma'\vdash x{:}\,K}$$

$$\frac{\Gamma_1,\Gamma_2\vdash J\ \ \Gamma_1,\Gamma_3\vdash \mathbf{valid}}{\Gamma_1,\Gamma_3,\Gamma_2\vdash J}(wkn)$$

(where $FV(\Gamma_2)\cap FV(\Gamma_3)=\emptyset$).

$$(2.1)\ \frac{\Gamma\vdash K\mathbf{kind}}{\Gamma\vdash K=K} \qquad (2.2)\ \frac{\Gamma\vdash K=K'}{\Gamma\vdash K'=K} \qquad (2.3)\ \frac{\Gamma\vdash K=K'\ \ \Gamma\vdash K'=K''}{\Gamma\vdash K=K''}$$

$$(2.4)\ \frac{\Gamma\vdash k{:}\,K}{\Gamma\vdash k=k{:}\,K} \qquad (2.5)\ \frac{\Gamma\vdash k=k'{:}\,K}{\Gamma\vdash k'=k{:}\,K} \qquad (2.6)\frac{\Gamma\vdash k=k'{:}\,K\ \ \Gamma\vdash k'=k''{:}\,K}{\Gamma\vdash k=k''{:}\,K}$$

$$(3.1)\frac{\Gamma\vdash k{:}\,K\ \ \Gamma\vdash K=K'}{\Gamma\vdash k{:}\,K'} \quad (3.2)\frac{\Gamma\vdash k=k'{:}\,K\ \ \Gamma\vdash K=K'}{\Gamma\vdash k=k'{:}\,K'}$$

$$(3.3)\frac{\Gamma,x{:}K,\Gamma'\vdash J\ \ \Gamma\vdash K=K'}{\Gamma,x{:}K',\Gamma'\vdash J}$$

$$(4.1)\frac{\Gamma\vdash \mathbf{valid}}{\Gamma\vdash \mathbf{Typekind}} \qquad (4.2)\frac{\Gamma\vdash A{:}\,\mathbf{Type}}{\Gamma\vdash El(A)\mathbf{kind}} \qquad (4.3)\frac{\Gamma\vdash A=B{:}\,\mathbf{Type}}{\Gamma\vdash El(A)=El(B)}$$

$$(5.1)\frac{\Gamma,x{:}K\vdash K'\mathbf{kind}}{\Gamma\vdash (x{:}K)K'\mathbf{kind}} \qquad (5.2)\frac{\Gamma,x{:}\,K_1\vdash K_1'=K_2'\ \ \Gamma\vdash K_1=K_2}{\Gamma\vdash (x{:}K_1)K_1'=(x{:}K_2)K_2'}$$

$$(5.3)\frac{\Gamma,x{:}\,K\vdash k{:}\,K'}{\Gamma\vdash [x{:}K]k{:}(x{:}K)K'} \qquad (5.4)\frac{\Gamma,x{:}\,K_1\vdash k_1=k_2{:}K\ \ \Gamma\vdash K_1=K_2}{\Gamma\vdash [x{:}K_1]k_1=[x{:}K_2]k_2{:}(x{:}K_1)K}$$

$$(5.5)\frac{\Gamma\vdash f{:}(x{:}K)K'\ \ \Gamma\vdash k{:}\,K}{\Gamma\vdash f(k){:}[k/x]K'} \quad (5.6)\frac{\Gamma\vdash f=f'{:}(x{:}K)K'\ \ \Gamma\vdash k_1=k_2{:}\,K}{\Gamma\vdash f(k_1)=f'(k_2){:}[k_1/x]K'}$$

$$(5.7)\frac{\Gamma,x{:}\,K\vdash k'{:}\,K'\ \ \Gamma\vdash k{:}\,K}{\Gamma\vdash ([x{:}K]k')k=[k/x]k'{:}[k/x]K'} \quad (5.8)\frac{\Gamma\vdash f{:}(x{:}K)K'\ \ x\notin FV(\Gamma)}{\Gamma\vdash [x{:}K]f(x)=f{:}(x{:}K)K'}$$

$$(6.1)\frac{\Gamma,x{:}\,K,\Gamma'\vdash \mathbf{valid}\ \ \Gamma\vdash k{:}\,K}{\Gamma,[k/x]\Gamma'\vdash \mathbf{valid}} \qquad (6.2)\frac{\Gamma,x{:}\,K,\Gamma'\vdash K'\mathbf{kind}\ \ \Gamma\vdash k{:}\,K}{\Gamma,[k/x]\Gamma'\vdash [k/x]K'\mathbf{kind}}$$

$$(6.3)\frac{\Gamma,x{:}\,K,\Gamma'\vdash k'{:}\,K'\ \ \Gamma\vdash k{:}\,K}{\Gamma,[k/x]\Gamma'\vdash [k/x]k'{:}[k/x]K'} \qquad (6.4)\ \frac{\Gamma,x{:}\,K,\Gamma'\vdash K'=K''\ \ \Gamma\vdash k{:}\,K}{\Gamma,[k/x]\Gamma'\vdash [k/x]K'=[k/x]K''}$$

$$(6.5)\frac{\Gamma,x{:}\,K,\Gamma'\vdash k'=k''{:}\,K'\ \ \Gamma\vdash k{:}\,K}{\Gamma,[k/x]\Gamma'\vdash [k/x]k'=[k/x]k''{:}\,K'}$$

$$(6.6)\ \frac{\Gamma,x{:}\,K,\Gamma'\vdash K'\mathbf{kind}\ \ \Gamma\vdash k_1=k_2{:}\,K}{\Gamma,[k_1/x]\Gamma'\vdash [k_1/x]K'=[k_2/x]K'} \qquad (6.7)\ \frac{\Gamma,x{:}\,K,\Gamma'\vdash k'{:}\,K'\ \ \Gamma\vdash k_1=k_2{:}\,K}{\Gamma,[k/x]\Gamma'\vdash [k_1/x]k'=[k_1/x]k'{:}[k/x]K'}$$

Figure 1: Inference rules of LF.