Combining Incoherent Coercions for Σ-types

Yong Luo and Zhaohui Luo

Department of Computer Science, University of Durham

Abstract. Coherence is a vital requirement for the correct use of coercive subtyping for abbreviation and other applications. However, some coercions are incoherent, although very useful. A typical example of such is the subtyping rules for Σ-types: the component-wise rules and the rule of the first projection. Both of these groups of rules are often used in practice (and coherent themselves), but they are incoherent when put together directly. In this paper, we study this case for Σ-types by introducing a new subtyping relation and the resulting system enjoys the properties of coherence and admissibility of substitution and transitivity.

1 Introduction

Coercive subtyping for dependent type theories, as studied in [Luo97,Luo99] and implemented in proof assistants such as Lego [LP92], Coq [B$^+$00] and Plastic [CL01], is a powerful abbreviation mechanism and has been used in applications of proof development (*e.g.* [Bai98]). An important requirement of coercion mechanism is that of coherence, that is, coercions between any two types must be computationally equal. This requirement is essential for the consistent use and correct implementation of the mechanism.

A coherence problem Some coercions cannot be put together directly in a coherent way, although very useful. A typical example of such coercions is those concerning Σ-types (types of dependent pairs). There are at least two sets of natural and useful coercion rules: the component-wise subtyping rules and the rule of the first projection. They are coherent separately (see [LL01] for the coherence of the former and [Bai98] for the use of the latter), but incoherent when put together directly (see the counter example in section 3 for details). This prevents them from being used together.

Our solution to this coherence problem is basically, in this paper, by introducing a new subtyping relation and giving a new formulation of coercive subtyping, to ensure that there is only one coercion (with respect to computational equality) between any two types (if there is one).

Transitivity This new formulation not only satisfies coherence requirements but also enjoys other properties, particularly, the admissibility of substitution and transitivity because such properties are important for an implementation of coercive subtyping. Through our investigation, we found out that the property of admissibility of transitivity is actually very hard to come by. In this paper, we shall consider two subtyping relations simultaneously, give new transitivity rules and prove that all of them are admissible.

In section 2, we shall give an overview of coercive subtyping and introduce some concepts such as Well-Defined Coercions and notations to be used later in the paper. In section 3, the coherence problem and its solution will be intuitively explained through a counter example. In section 4, there is a formal presentation of the solution. A new definition of coherence, new rules of substitution and transitivity are also given. Some important properties, coherence and the admissibility of substitution and transitivity, are proved. Discussions are in the last section, where we discuss issues such as decidability and wider applications of the methods developed in this paper.

2 Coercive subtyping and well-defined coercions

In this section, we give an overview of coercive subtyping, introduce some notations and the concept of well-defined coercions that will be used later in the paper.

2.1 Coercive subtyping

The basic idea of coercive subtyping, as studied in [Luo99], 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 object of type B via c, where c is a functional operation from A to B in the type theory.

A coercion plays the role of abbreviation. More precisely, if c is a coercion from K_0 to K, then a functional operation f with domain K can be applied to any object k_0 of K_0 and the application $f(k_0)$ is 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 the logical framework, becomes very powerful. The second author and his colleagues have developed the framework of coercive subtyping that covers variety of subtyping relations including those represented by parameterised coercions and coercions between parameterised inductive types. See [Luo99,Bai98,CL01,LC98,CLP01] for details of some of these development and applications of coercive subtyping.

Some important meta-theoretic aspects of coercive subtyping have been studied. In particular, the results on conservativity and on transitivity elimination for subkinding have been proved in [JLS98,SL02]. The main result of [SL02] is essentially that coherence of basic subtyping rules does imply conservativity. These results not only justify the adequacy of the theory from the proof-theoretic consideration, but also provide the proof-theoretic basis for implementation of coercive subtyping.

How to prove coherence and admissibility of transitivity at the type level has been studied in [LL01] recently. In particular, the concept of *Well-defined* *coercions* has been developed, and the suitable subtyping rules for Π-types and Σ-types have been given as examples to demonstrate these proof techniques.

Coercive subtyping is formally formulated as an extension of (type theories specified in) the logical framework LF[1], whose rules are given in [Luo94]. Types in LF are called kinds. The kind $Type$ represents the conceptual universe of types and a kind of form $(x : K)K'$ represents the dependent product with functional operations f as objects (*e.g.*, abstraction $[x : K]k'$) which can be applied to objects of kind K to form application $f(k)$. For every type (an object of kind $Type$), $El(A)$ is the kind of objects of A. LF can be used to specify type theories, such as Martin-Löf's type theory [NPS90] and UTT [Luo94].

As presented in [Luo99], a system with coercive subtyping is an extension of any type theory specified in LF by a set of basic subtyping rules $\mathcal{R}$ whose conclusions are subtyping judgements of the form $\Gamma \vdash A <_c B : Type$. And the subtyping rules in $\mathcal{R}$ are supposed to be coherent.

Notation We shall use the following notations:

- We sometimes use $M[x]$ to indicate that variable x may occur free in M.
- Context equality: for $\Gamma \equiv x_1 : K_1, ..., x_n : K_n$ and $\Gamma' \equiv x_1 : K'_1, ..., x_n : K'_n$, we shall write $\vdash \Gamma = \Gamma'$ for the sequence of judgements $\vdash K_1 = K'_1$, ..., $x_1 : K_1, ..., x_{n-1} : K_{n-1} \vdash K_n = K'_n$.
- Types of non-dependent pairs: if A and B are types, we sometimes write $A \times B$ for $\Sigma(A, [x : A]B)$ where x is not free in B.

2.2 Well-defined Coercions

Recently, a new concept of Well-defined Coercions (WDC) has been developed in [LL01]. Suppose there is a set of coercions, which is coherent and have admissibility properties, we prove that, after adding new subtyping rules, the extended system still keeps the coherence and admissibility properties.

Definition 1. (Well-defined coercions) *If $\mathcal{C}$ is a set of subtyping judgements of the form $\Gamma \vdash M <_d M' : Type$ which satisfies the following conditions, we say that $\mathcal{C}$ is a* well-defined set of judgements for coercions*, or briefly called* Well-defined Coercions (WDC)*.*

1. *(Coherence)*
   (a) *$\Gamma \vdash A <_c B : Type \in \mathcal{C}$ implies $\Gamma \vdash A : Type$, $\Gamma \vdash B : Type$ and $\Gamma \vdash c : (A)B$.*
   (b) *$\Gamma \vdash A <_c A : Type \notin \mathcal{C}$ for any Γ, A, and c.*
   (c) *$\Gamma \vdash A <_{c_1} B : Type \in \mathcal{C}$ and $\Gamma \vdash A <_{c_2} B : Type \in \mathcal{C}$ imply $\Gamma \vdash c_1 = c_2 : (A)B$.*
2. *(Congruence) $\Gamma \vdash A <_c B : Type \in \mathcal{C}$, $\Gamma \vdash A = A' : Type$, $\Gamma \vdash B = B' : Type$ and $\Gamma \vdash c = c' : (A)B$ imply $\Gamma \vdash A' <_{c'} B' \in \mathcal{C}$.*
3. *(Transitivity) $\Gamma \vdash A <_{c_1} B : Type \in \mathcal{C}$ and $\Gamma \vdash B <_{c_2} A' : Type \in \mathcal{C}$ imply $\Gamma \vdash A <_{c_2 \circ c_1} A' : Type \in \mathcal{C}$.*

[1] The LF here is different from the Edinburgh Logical Framework [HHP87].

4. *(Substitution)* $\Gamma, x : K, \Gamma' \vdash A <_c B : Type \in \mathcal{C}$ *implies for any* k *such that* $\Gamma \vdash k : K$, $\Gamma, [k/x]\Gamma' \vdash [k/x]A <_{[k/x]c} [k/x]B : Type \in \mathcal{C}$.
5. *(Weakening)* $\Gamma \vdash A <_c B : Type \in \mathcal{C}$, $\Gamma \subseteq \Gamma'$ *and* Γ' *is valid imply* $\Gamma' \vdash A <_c B : Type \in \mathcal{C}$.

In this paper, the set $\mathcal{R}$ of basic coercion rules includes the following rule, where $\mathcal{C}$ is a WDC:

$$(WDCrule) \qquad \frac{\Gamma \vdash A <_c B : Type \in \mathcal{C}}{\Gamma \vdash A <_c B : Type}$$

3 The basic subtyping rules and the coherence problem

In this section, we give an example to illustrate the coherence problem of the component-wise subtyping rules for Σ-types and the subtyping rule of its first projection and explain informally the solution through a counter example.

Subtyping rules for Σ-types As studied in [LL01], there are three component-wise subtyping rules for Σ-types. One of these rules is the following.

$$(First\,Component\,rule) \quad \frac{\Gamma \vdash A <_c A' : Type \quad \Gamma \vdash B : (A')Type}{\Gamma \vdash \Sigma(A, B \circ c) <_{d_1} \Sigma(A', B) : Type}$$

where $d_1 = [z : \Sigma(A, B \circ c)]pair(A', B, c(\pi_1(A, B \circ c, z)), \pi_2(A, B \circ c, z))$, which basically means that, for example, $A \times B$ is a subtype of $A' \times B$ if A , A' and B are types and A is a subtype of A'.

The coercion of the first projection is very useful; for example, it is used significantly in Bailey's PhD thesis [Bai98] for formalisation of mathematics. Formally, the subtyping rule is the following:

$$(\pi_1 rule) \qquad \frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : (A)Type}{\Gamma \vdash \Sigma(A, B) <_{\pi_1(A,B)} A : Type}$$

With this coercion, it is very easy to express some mathematical properties. For example, the type of collection of groups is a subtype of the type of semi-groups (*i.e.* a group is also a semi-group). Any functional operator with the domain of semi-groups can be applied to any group with a coercion.

A counter example If the subtyping rule $(\pi_1 rule)$ and the component-wise subtyping rules for Σ-types are combined together, we would have the following two derivations.

The first derivation is

$$\frac{\dfrac{\Gamma \vdash A : Type \; \Gamma \vdash B : (A)Type}{\Gamma \vdash \Sigma(A, B) : Type} \quad \dfrac{\Gamma \vdash B : (A)Type}{\Gamma \vdash B \circ \pi_1(A, B) : (\Sigma(A, B))Type}}{\Gamma \vdash \Sigma(\Sigma(A, B), B \circ \pi_1(A, B)) <_{d_1} \Sigma(A, B) : Type}$$

and the rule $(\pi_1 rule)$ is used in the last step.

The second derivation is

$$\dfrac{(\pi_1 rule)\,\dfrac{\Gamma \vdash A : Type \; \Gamma \vdash B : (A)Type}{\Gamma \vdash \Sigma(A,B) <_{\pi_1(A,B)} A : Type} \quad \Gamma \vdash B : (A)Type}{\Gamma \vdash \Sigma(\Sigma(A,B), B \circ \pi_1(A,B)) <_{d_2} \Sigma(A,B) : Type}$$

and the rule $(\pi_1 rule)$ is used in the first step and the First Component rule is used in the last step.

There are two coercions d_1 and d_2 from type $\Sigma(\Sigma(A,B), B \circ \pi_1(A,B))$ to type $\Sigma(A,B)$[2] and we have the following equations

$$d_1(pair(pair(a,b_1),b_2)) = pair(a,b_1)$$
$$d_2(pair(pair(a,b_1),b_2)) = pair(a,b_2)$$

We can see that d_1 and d_2 are neither computationally nor extensionally equal. Hence, the vital requirement of coercive subtyping system, coherence, fails.

Informal explanation of our solution From the above counter example, we see that the existence of the two derivations makes the system incoherent. To make it coherent, a natural way is to block one of the derivations. The first one cannot be blocked, otherwise we lose the meaning that the first projection (π_1) is regarded as coercion. And hence we can only block the second derivation. More precisely, we must not allow $\Gamma \vdash A <_c A' : Type$ is used as the first promise of the component-wise subtyping rules if it is (directly) derived from $\pi_1 rule$. In other words, a condition of the component-wise subtyping rules is that the first promise is not (directly) derived from $\pi_1 rule$. There are several attempts to satisfy this condition, one of which is to consider a notion of size as a side-condition because A is a sub-term of $\Sigma(A,B)$ in the conclusion of $\pi_1 rule$, and their sizes are intuitively different. However, the well-definedness of size is problematic when we present the whole subtyping system (see discussion section for more details).

In this paper, rather than thinking of any side-conditions, we introduce a new subtyping relation $(\prec)$ to represent coercion π_1. This new subtyping relation will never appear in the first premises of the component-wise subtyping rules and hence the unwanted derivations such as the second one in the counter example are blocked.

To make the subtyping system coherent is one thing; to make it also enjoy the property of admissibility of transitivity is another. During our investigation, we experienced that some formulations satisfy the property of coherence, but not the admissibility of transitivity. The formulation in the next section will enjoy all these properties.

[2] There are two different coercions from $(A \times B) \times B$ to $A \times B$ if A and B are types.

4 A formal presentation

In this section, we shall give a formal presentation of a new subtyping relation and related subtyping rules. The coherence and admissibility of substitution and transitivity will also be proved.

4.1 A new subtyping relation

We have seen the problem with the combination of the component-wise subtyping rules and the subtyping rule of the first projection. Now, we introduce a new relation to solve this problem and, consider a new system $T[\mathcal{R}\pi_1]$, which is an extension of coercive subtyping with the judgement form:

- $\Gamma \vdash A \prec_c B : Type$ asserts that type A is a subtype of type B with c.

As we will see later, subtyping relation $<$ and $\prec$ are different. $\prec$ represents the idea that π_1 is regarded as a coercion, but $<$ doesn't.

The coercive definition rules The main idea of coercive subtyping can informally be represented by the following coercive definition rule (contexts are omitted):

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

The same idea is for the new subtyping relation. A new basic subkinding rule for $\prec$ is the following:

$$\frac{A \prec_c B : Type}{El(A) <_c El(B)}$$

By the coercive definition rule, we have the following derivable rule:

$$\frac{A \prec_c B : Type \quad k : El(A) \quad f : (x : El(B))K}{f(k) = f(c(k)) : [c(k)/x]K}$$

which says that if $A \prec_c B$, any functional operator f with domain B can be applied to any object x of A and, $f(x) = f(c(x))$.

We present the new subtyping system in two stages: first an intermediate system $T[\mathcal{R}\pi_1]_0$ and the definition of coherence, and then the system $T[\mathcal{R}\pi_1]$.

4.2 The system $T[\mathcal{R}\pi_1]_0$ and $T[\mathcal{R}\pi_1]$

Formally, $T[\mathcal{R}\pi_1]_0$ is an extension of type theory T (only) with the following rules:

- A set $\mathcal{R}$ of basic subtyping rules whose conclusions are subtyping judgements of the form $\Gamma \vdash A <_c B : Type$.

- The following congruence rule for subtyping judgements

$$(Cong)\qquad \frac{\Gamma \vdash A <_c B : Type \quad \Gamma \vdash A = A' : Type \ \Gamma \vdash B = B' : Type \ \Gamma \vdash c = c' : (A)B}{\Gamma \vdash A' <_{c'} B' : Type}$$

- The new subtyping rules for the first projection in Figure 1, whose conclusions are of the form $\Gamma \vdash A \prec_c B : Type$.

Notation: we shall use $\Gamma \vdash A \propto_c B : Type$ to represent $\Gamma \vdash A <_c B : Type$ or $\Gamma \vdash A \prec_c B : Type$. For example, $\frac{\Gamma \vdash A \propto_c B : Type}{J}$ actually represents two rules $\frac{\Gamma \vdash A <_c B : Type}{J}$ and $\frac{\Gamma \vdash A \prec_c B : Type}{J}$; $\frac{\Gamma \vdash A \propto_c B : Type \quad \Gamma' \vdash A' \propto_{c'} B' : Type}{J}$ actually represents four rules. We shall also say that A is a subtype of B or there is a coercion c from A to B if $\Gamma \vdash A \propto_c B : Type$.

New subtyping rule for the first projection:

$$\frac{\Gamma \vdash A : Type \quad \Gamma \vdash B : (A)Type}{\Gamma \vdash \Sigma(A, B) \prec_{\pi_1(A,B)} A : Type}$$

$$\frac{\Gamma \vdash A \propto_c A' : Type \quad \Gamma \vdash B : (A)Type}{\Gamma \vdash \Sigma(A, B) \prec_{c \circ \pi_1(A,B)} A' : Type}$$

New congruence rule:

$$\frac{\Gamma \vdash A \prec_c B : Type \quad \Gamma \vdash A = A' : Type \ \Gamma \vdash B = B' : Type \ \Gamma \vdash c = c' : (A)B}{\Gamma \vdash A' \prec_{c'} B' : Type}$$

Fig. 1. New subtyping rules for the first projection

Remark 1. We have the following remarks.

- The basic understanding of the new subtyping rules for the first projection is that $\Sigma(A, B)$ is a subtype of A' if $A = A'$ or A is a subtype of A'.
- New substitution and transitivity rules for subtyping relations $<$ and $\prec$ will be given later and, we will prove that all of them are admissible. We do not include them in $T[\mathcal{R}\pi_1]_0$.

New subtyping rules for parameterised inductive types Now, we give the component-wise subtyping rules for Σ-types and the rules for Π-types in Figure 2 and 3 to demonstrate what the subtyping rules should be for the new subtyping relation.

First Component rule:

$$\frac{\Gamma \vdash A <_c A' : Type \;\; \Gamma \vdash B : (A')Type}{\Gamma \vdash \Sigma(A, B \circ c) <_{d_1} \Sigma(A', B) : Type}$$

where $d_1 = [z : \Sigma(A, B \circ c)]pair(A', B, c(\pi_1(A, B \circ c, z)), \pi_2(A, B \circ c, z))$

Second Component rule:

$$\frac{\Gamma \vdash B : (A)Type \;\; \Gamma \vdash B' : (A)Type \;\; \Gamma, x : A \vdash B(x) \propto_{e[x]} B'(x) : Type}{\Gamma \vdash \Sigma(A, B) <_{d_2} \Sigma(A, B') : Type}$$

where $d_2 = [z : \Sigma(A, B)]pair(A, B', \pi_1(A, B, z), e[\pi_1(A, B, z)](\pi_2(A, B, z)))$

First-Second Component rule:

$$\frac{\begin{array}{c}\Gamma \vdash A <_c A' : Type \; \Gamma \vdash B : (A)Type \; \Gamma \vdash B' : (A')Type \\ \Gamma, x : A \vdash B(x) \propto_{e[x]} B'(c(x)) : Type\end{array}}{\Gamma \vdash \Sigma(A, B) <_{d_3} \Sigma(A', B') : Type}$$

where $d_3 = [z : \Sigma(A, B)]pair(A', B', c(\pi_1(A, B, z)), e[\pi_1(A, B, z)](\pi_2(A, B, z)))$

Fig. 2. New component-wise subtyping rules for Σ-types

Domain rule:

$$\frac{\Gamma \vdash A' \propto_c A : Type \;\; \Gamma \vdash B : (A)Type}{\Gamma \vdash \Pi(A, B) <_{d_1} \Pi(A', B \circ c) : Type}$$

where $d_1 = [f : \Pi(A, B)]\lambda(A', B \circ c, app(A, B, f) \circ c)$

Codomain rule:

$$\frac{\Gamma \vdash B : (A)Type \;\; \Gamma \vdash B' : (A)Type \;\; \Gamma, x : A \vdash B(x) \propto_{e[x]} B'(x) : Type}{\Gamma \vdash \Pi(A, B) <_{d_2} \Pi(A, B') : Type}$$

where $d_2 = [f : \Pi(A, B)]\lambda(A, B', [x : A]e[x](app(A, B, f, x)))$

Domain-Codomain rule:

$$\frac{\begin{array}{c}\Gamma \vdash A' \propto_c A : Type \; \Gamma \vdash B : (A)Type \; \Gamma \vdash B' : (A')Type \\ \Gamma, x' : A' \vdash B(c(x')) \propto_{e[x']} B'(x') : Type\end{array}}{\Gamma \vdash \Pi(A, B) <_{d_3} \Pi(A', B') : Type}$$

where $d_3 = [f : \Pi(A, B)]\lambda(A', B', [x' : A']e[x'](app(A, B, f, c(x'))))$

Fig. 3. New subtyping rules for Π-types

Remark 2. We have the following remarks.

- In Figure 2 and 3, the conclusions of the rules are always of the form $\Gamma \vdash A <_c B : Type$, no matter the premises are of the form $\Gamma \vdash A <_c B : Type$ or $\Gamma \vdash A \prec_c B : Type$.
- The essence of the new subtyping relation is that, the judgement form $\Gamma \vdash A \prec_c B : Type$ is never used in the premises of the first component of the component-wise subtyping rules in Figure 2. And hence the second derivation of the counter example in section 3 is blocked.
- The basic understanding of the new subtyping rules for Π-types is that $\Pi(A, B)$ is a subtype of $\Pi(A', B')$ if A' is a subtype of A and B is a sub-family of B' (we omit other cases such as: $\Pi(A, B)$ is a subtype of $\Pi(A, B')$ if B is a sub-family of B').
- For the new component-wise subtyping rules for Σ-types, because of the incoherence when π_1 is also regarded as a coercion, we need to have a stricter understanding, that is, $\Sigma(A, B)$ is a subtype of $\Sigma(A', B')$ if A is a subtype of A' and B is a sub-family of B' and the sizes of A and A' are the same ($size$ is defined in the definition 4). In the following section, we will prove that the sizes of A and B are the same if $\Gamma \vdash A <_c B : Type$ and, the size of A is bigger than the size of B if $\Gamma \vdash A \prec_c B : Type$.

The subtyping system we presented here covers all the coercions derived from the component-wise subtyping rules and the subtyping rule for the first projection when they are used separately. Actually, it has more coercions. For example, if A, B and C are different types, we can have a coercion from $A \times (B \times C)$ to $A \times B$ because there is a coercion from $B \times C$ to B. But we can never derive a coercion from $A \times (B \times C)$ to $A \times B$ by the component-wise subtyping rules or the subtyping rule for the first projection separately. What we have excluded are those coercions that need component-wise subtyping rules for Σ-types but the sizes of their first components are different. For example, we don't have a coercion from $(A \times B) \times C$ to $A \times C$ because the sizes of $A \times B$ and A are different although there is a coercion from $A \times B$ to A.

In $T[\mathcal{R}\pi_1]_0$, the subtyping judgements do not contribute to any derivation of a judgement of any other forms in the original type theory T. Therefore, we have the following lemma.

Lemma 1. *$T[\mathcal{R}\pi_1]_0$ is a conservative extension of T.*

Remark 3. As the two subtyping relations $<$ and $\prec$ do contribute to each other, $T[\mathcal{R}\pi_1]_0$ is not a conservative extension of $T[\mathcal{R}]_0$ whose subtyping judgements are only of the form $\Gamma \vdash A <_c B : Type$ (see [Luo99] for details).

Now, we define the most basic requirement for the new subtyping relation in the following.

Definition 2. (Coherence condition of $T[\mathcal{R}\pi_1]_0$) *We say that $T[\mathcal{R}\pi_1]_0$ is coherent if it has the following properties.*

1. $\Gamma \vdash A \propto_c B : Type$ *implies* $\Gamma \vdash A : Type$, $\Gamma \vdash B : Type$, *and* $\Gamma \vdash c : (A)B$.
2. $\Gamma \vdash A \propto_c B : Type$ *implies* $\Gamma \not\vdash A = B : Type$.
3. $\Gamma \vdash A <_c B : Type$ *and* $\Gamma \vdash A <_{c'} B : Type$ *imply* $\Gamma \vdash c = c' : (A)B$.
4. $\Gamma \vdash A \prec_c B : Type$ *and* $\Gamma \vdash A \prec_{c'} B : Type$ *imply* $\Gamma \vdash c = c' : (A)B$.
5. *(Disjointness)* $\Gamma \vdash A <_c B : Type$ *implies* $\Gamma \not\vdash A \prec_{c'} B : Type$ *for any* c', *and vice versa,* $\Gamma \vdash A \prec_c B : Type$ *implies* $\Gamma \not\vdash A <_{c'} B : Type$ *for any* c'.

Remark 4. One may consider a more general coherence condition like, if $\Gamma \vdash A \propto_c B : Type$ and $\Gamma \vdash A \propto_{c'} B : Type$ then $\Gamma \vdash c = c' : (A)B$. This will include the case which both $\Gamma \vdash A <_c B : Type$ and $\Gamma \vdash A \prec_c B : Type$ may happen. However, one of the reasons we need the new subtyping relation ($\prec$) is deliberately to make sure that $\Gamma \vdash A <_c B : Type$ and $\Gamma \vdash A \prec_c B : Type$ may never hold at the same time for any A and B. Disjointness is regarded as a part of coherence condition.

The system $T[\mathcal{R}\pi_1]$ The system $T[\mathcal{R}\pi_1]$ is an extension of $T[\mathcal{R}\pi_1]_0$ with the inference rules in Appendix. Comparing with the original subkinding rules in [Luo99], a new rule is added.

$$(New\ Basic\ Subkinding\ Rule) \qquad \frac{\Gamma \vdash A \prec_c B : Type}{\Gamma \vdash El(A) <_c El(B)}$$

There is only one subkinding judgement form $\Gamma \vdash K <_c K'$, although there are two subtyping judgement forms $\Gamma \vdash A <_c B : Type$ and $\Gamma \vdash A \prec_c B : Type$. At the kind level, we are more concerned with the existence of a coercion no matter it is derived from which form at the type level.

Remark 5. The main result of [SL02] is essentially that coherence of subtyping rules does imply conservativity. In the next section, we shall also prove the coherence of $T[\mathcal{R}\pi_1]_0$. So, $T[\mathcal{R}\pi_1]$ is also expected to be a conservative extension of T.

4.3 Coherence of $T[\mathcal{R}\pi_1]_0$

Now, we prove the coherence of $T[\mathcal{R}\pi_1]_0$, which essentially says that coercions between any two types must be unique. In this paper, the set $\mathcal{R}$ of basic subtyping consists of the rule ($WDCrule$) and the new subtyping rules for Σ-types and Π-types (in Figure 2 and 3) and, the system $T[\mathcal{R}\pi_1]_0$ also includes the congruence rule ($Cong$) and the new subtyping rules in Figure 1. Furthermore, we assume that for any judgement $\Gamma \vdash A <_c B : Type \in \mathcal{C}$, neither A nor B is computationally equal to a Σ-type or Π-type. We also assume that the original type theory T has good properties, in particular the properties of Church-Rosser and Strong Normalisation and the property of context replacement by equal kinds.

We give a definition of $size(A)$ that only counts how many times that π_1 can be applied for an object of type A. In order to define $size$, we define $presize$ first.

Definition 3. ***(presize)*** *Let $\Gamma \vdash M : Type$ be a derivable judgement in $T[\mathcal{R}\pi_1]_0$ and M a normal form (*i.e. *$M \equiv nf(M)$),*

1. *if M is not a Σ-type then $presize(M) =_{df} 0$,*
2. *if $M \equiv \Sigma(A, B)$ then $presize(M) =_{df} presize(A) + 1$.*

Remark 6. For the second case, because M is a normal form, so is A. Therefore $presize$ is well-defined.

Definition 4. ***(size)*** *The definition of size in $T[\mathcal{R}\pi_1]_0$: Let $\Gamma \vdash M : Type$ be a derivable judgement in $T[\mathcal{R}\pi_1]_0$, $size(M) =_{df} presize(nf(M))$.*

Remark 7. $T[\mathcal{R}\pi_1]_0$ is a conservative extension of T and every well-typed term in T has its unique normal form. So, the value of $size(M)$ is unique and $size$ is well-defined.

Lemma 2. *In $T[\mathcal{R}\pi_1]_0$, if $\Gamma \vdash M_1 = M_2 : Type$ then $size(M_1) = size(M_2)$.*

Proof. $T[\mathcal{R}\pi_1]_0$ is a conservative extension of T and T has properties of Church-Rosser and strong normalisation, *i.e.* $nf(M_1) \equiv nf(M_2)$.

Lemma 3. *Let $\Gamma \vdash M : Type$ be a derivable judgement in $T[\mathcal{R}\pi_1]_0$.*

- *if M is not computationally equal to a Σ-type then $size(M) = 0$ and,*
- *if $\Gamma \vdash M = \Sigma(A, B) : Type$ then $size(M) = size(A) + 1$.*

Proof. By the definition of $size$ and Lemma 2.

Lemma 4. *In $T[\mathcal{R}\pi_1]_0$, if $\Gamma \vdash M_1 <_d M_2 : Type$ then $size(M_1) = size(M_2)$.*

Proof. By induction on derivations and Lemma 2 and Lemma 3. Note that $size(M_1) = size(M_2) = 0$ if the last rule of $\Gamma \vdash M_1 <_d M_2 : Type$ is one of the rules for Π-types.

Lemma 5. *In $T[\mathcal{R}\pi_1]_0$, if $\Gamma \vdash M_1 \prec_c M_2 : Type$ then $size(M_1) > size(M_2)$.*

Proof. By induction on derivations and Lemma 2, Lemma 3 and Lemma 4.

The following theorems prove the coherence of $T[\mathcal{R}\pi_1]_0$.

Theorem 1.
- *If $\Gamma \vdash M_1 \propto_c M_2 : Type$ then $\Gamma \vdash M_1 : Type$, $\Gamma \vdash M_2 : Type$ and $\Gamma \vdash c : (M_1)M_2 : Type$.*
- *If $\Gamma \vdash M_1 \propto_c M_2 : Type$ then $\Gamma \not\vdash M_1 = M_2 : Type$.*
- *If $\Gamma \vdash M_1 \prec_c M_2 : Type$ then $\Gamma \not\vdash M_1 <_d M_2 : Type$ for any d. And vice versa, if $\Gamma \vdash M_1 <_c M_2 : Type$, then $\Gamma \not\vdash M_1 \prec_d M_2 : Type$ for any d.*

Proof. By induction on derivations, the definition of WDC, Lemma 4 and Lemma 5.

Theorem 2. *If $\vdash \Gamma = \Gamma'$ $\Gamma \vdash M_1 = M_1' : Type$ and $\Gamma \vdash M_2 = M_2' : Type$ and*

1. *$\Gamma \vdash M_1 <_d M_2 : Type$ and $\Gamma' \vdash M_1' <_{d'} M_2' : Type$, or*

2. $\Gamma \vdash M_1 \prec_d M_2 : Type$ *and* $\Gamma' \vdash M_1' \prec_{d'} M_2' : Type$

then $\Gamma \vdash d = d' : (M_1)M_3$.

Proof. By induction on derivations. A most important arguement in this proof is that, any derivations of $\Gamma \vdash M_1 <_d M_2$ and $\Gamma' \vdash M_1' <_d M_2'$, or $\Gamma \vdash M_1 \prec_d M_2$ and $\Gamma' \vdash M_1' \prec_{d'} M_2'$ must contain sub-derivations whose last rules are the same rule, followed by applications of the congruence rules.

4.4 Admissibility of substitution and transitivity

Now, we give the subtyping rules of substitution and transitivity and, prove that these rules are admissible. In an implementation of coercive subtyping, these rules are ignored simply because they cannot be directly implemented. For this reason among others, proving the admissibility of such rules (or their elimination) is always an important task for any subtyping system.

Admissible substitution rules The substitution rules are as follows, which are what we expect normally.

$$\frac{\Gamma, x : K, \Gamma' \vdash A <_c B : Type \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]A <_{[k/x]c} [k/x]B : Type}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash A \prec_c B : Type \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]A \prec_{[k/x]c} [k/x]B : Type}$$

Admissible transitivity rules We give the following four transitivity rules that are basically saying that if there are coercions c and c' from type A to B and from type B to C, then $c' \circ c$ is a coercion from type A to C.

$$\frac{\Gamma \vdash A <_{c_1} B : Type \quad \Gamma \vdash B <_{c_2} C : Type}{\Gamma \vdash A <_{c_2 \circ c_1} C : Type} \qquad \frac{\Gamma \vdash A \prec_{c_1} B : Type \quad \Gamma \vdash B \prec_{c_2} C : Type}{\Gamma \vdash A \prec_{c_2 \circ c_1} C : Type}$$

$$\frac{\Gamma \vdash A <_{c_1} B : Type \quad \Gamma \vdash B \prec_{c_2} C : Type}{\Gamma \vdash A \prec_{c_2 \circ c_1} C : Type} \qquad \frac{\Gamma \vdash A \prec_{c_1} B : Type \quad \Gamma \vdash B <_{c_2} C : Type}{\Gamma \vdash A \prec_{c_2 \circ c_1} C : Type}$$

Remark 8. The above transitivity rule are sufficient and correct, in the sense that, first, they capture the meaning of transitivity, and second, they enjoy the properties in the lemmas 4 and 5 . Other rules of different combination such as the rule

$$\frac{\Gamma \vdash A <_{c_1} B : Type \quad \Gamma \vdash B <_{c_2} C : Type}{\Gamma \vdash A \prec_{c_2 \circ c_1} C : Type}$$

are not correct and contradictory to the above properties.

Theorem 3. ***(Substitution in*** $T[\mathcal{R}\pi_1]_0$***)*** *If* $\Gamma \vdash k : K$ *and*

1. *if* $\Gamma, x : K, \Gamma' \vdash M_1 <_c M_2 : Type$, *then* $\Gamma, [k/x]\Gamma' \vdash [k/x]M_1 <_{[k/x]c} [k/x]M_2 : Type$, *and*

2. *if* $\Gamma, x : K, \Gamma' \vdash M_1 \prec_c M_2 : Type$, *then* $\Gamma, [k/x]\Gamma' \vdash [k/x]M_1 \prec_{[k/x]c} [k/x]M_2 : Type$.

Proof. By induction on derivations.

In order to prove the admissibility of the transitivity rules, we also need to prove the theorem about weakening.

Theorem 4. ***(Weakening in*** $T[\mathcal{R}\pi_1]_0$***)*** *If* $\Gamma \subseteq \Gamma'$, Γ' *is valid and*

1. *if* $\Gamma \vdash M_1 <_c M_2 : Type$ *then* $\Gamma' \vdash M_1 <_c M_2 : Type$, *and*
2. *if* $\Gamma \vdash M_1 \prec_c M_2 : Type$ *then* $\Gamma' \vdash M_1 \prec_c M_2 : Type$.

Proof. By induction on derivations.

To prove the admissibility of transitivity rules, the usual methods (*e.g.* by induction on derivations) do not seem to work. We develop a new measure (*Depth*) that is an adoption of the measure (*depth*) developed by Chen, Aspinall and Companoni [Che98]. In the measure *Depth*, the subtyping judgements ($<$ and $\prec$) only count.

Definition 5. *(Depth) Let* D *be a derivation of a subtyping judgement of the form* $\Gamma \vdash A <_c B : Type$ *or* $\Gamma \vdash A \prec_c B : Type$.

$$D : \qquad \frac{S_1 \; ... \; S_n \; T_1 \; ... \; T_m}{\Gamma \vdash A \propto_c B : Type}$$

where $\Gamma \vdash A \propto_c B : Type$ *represents* $\Gamma \vdash A <_c B : Type$ *or* $\Gamma \vdash A \prec_c B : Type$, S_1,...,S_n *are derivations of subtyping judgements of the form* $\Gamma \vdash M_1 <_d M_2 : Type$ *or* $\Gamma \vdash M_1 \prec_d M_2 : Type$ *and,* T_1,..., T_m *are derivations of other forms of judgements,*

$$Depth(D) =_{df} 1 + max\{Depth(S_1), ..., Depth(S_n)\}$$

Specially, if $n = 0$ *then* $Depth(D) =_{df} 1$.

The following lemmas show that, from a derivation D of a subtyping judgement J one can always get a derivation D' of the judgement obtained from J by context replacement such that D and D' have the same depth.

Lemma 6. *If* $\vdash \Gamma = \Gamma'$ *and*

1. *if* D *is a derivation of* $\Gamma \vdash M_1 <_d M_2 : Type$, *then there is a derivation* D' *of* $\Gamma' \vdash M_1 <_d M_2 : Type$ *such that* $Depth(D) = Depth(D')$, *or*
2. *if* D *is a derivation of* $\Gamma \vdash M_1 \prec_d M_2 : Type$, *then there is a derivation* D' *of* $\Gamma' \vdash M_1 \prec_d M_2 : Type$ *such that* $Depth(D) = Depth(D')$.

Proof. By induction on derivations.

Lemma 7. *If* $\Gamma, x : K, \Gamma' \vdash M_1 <_{c_1} M_2 : Type \in \mathcal{C}$ *and* $\Gamma \vdash c_2 : (K')K$ *then* $\Gamma, y : K', [c_2(y)/x]\Gamma' \vdash [c_2(y)/x]M_1 <_{[c_2(y)/x]c_1} [c_2(y)/x]M_2 : Type \in \mathcal{C}$.

Proof. By the weakening and substitution in the definition of WDC.

Lemma 8. *If $\Gamma \vdash c_2 : (K')K$ and,*

1. *if D is a derivation of $\Gamma, x : K, \Gamma' \vdash M_1 <_{c_1} M_2 : Type$, then there is a derivation D' of $\Gamma, y : K', [c_2(y)/x]\Gamma' \vdash [c_2(y)/x]M_1 <_{[c_2(y)/x]c_1} [c_2(y)/x]M_2 : Type$ such that $Depth(D) = Depth(D')$, or*
2. *if D is a derivation of $\Gamma, x : K, \Gamma' \vdash M_1 <_{c_1} M_2 : Type$, then there is a derivation D' of $\Gamma, y : K', [c_2(y)/x]\Gamma' \vdash [c_2(y)/x]M_1 <_{[c_2(y)/x]c_1} [c_2(y)/x]M_2 : Type$ such that $Depth(D) = Depth(D')$.*

Proof. By induction on derivations and Lemma 7. The theorem of weakening and substitution in type theory T and the property of conservativity of $T[\mathcal{R}\pi_1]_0$ over T are also needed in this proof.

Now, we can prove the admissibility of transitivity rules.

Theorem 5. ***(Transitivity in** $T[\mathcal{R}\pi_1]_0$**)*** *If $\Gamma \vdash M_2 = M_2' : Type$ and*

1. *if $\Gamma \vdash M_1 <_{d_1} M_2 : Type$ and $\Gamma \vdash M_2' <_{d_2} M_3 : Type$, then $\Gamma \vdash M_1 <_{d_2 \circ d_1} M_3 : Type$, and*
2. *$\Gamma \vdash M_1 \prec_{d_1} M_2 : Type$ and $\Gamma \vdash M_2' \prec_{d_2} M_3 : Type$, then $\Gamma \vdash M_1 \prec_{d_2 \circ d_1} M_3 : Type$.*
3. *if $\Gamma \vdash M_1 <_{d_1} M_2 : Type$ and $\Gamma \vdash M_2' \prec_{d_2} M_3 : Type$, then $\Gamma \vdash M_1 \prec_{d_2 \circ d_1} M_3 : Type$, and*
4. *$\Gamma \vdash M_1 \prec_{d_1} M_2 : Type$ and $\Gamma \vdash M_2' <_{d_2} M_3 : Type$, then $\Gamma \vdash M_1 \prec_{d_2 \circ d_1} M_3 : Type$, and*

Proof. By induction on $Depth(D) + Depth(D')$, where D is a derivation of $\Gamma \vdash M_1 <_{d_1} M_2 : Type$ or $\Gamma \vdash M_1 \prec_{d_1} M_2 : Type$, D' is a derivation of $\Gamma \vdash M_2' <_{d_2} M_3 : Type$ or $\Gamma \vdash M_2' \prec_{d_2} M_3 : Type$.

5 Discussions

Side conditions[3] In order to block the unwanted derivations, one may still try to keep the rule $\pi_1 rule$ in section 3 and use side conditions for the First Component rule, without introducing any new subtyping relation. For instance, one of such side conditions for the First Component rule is the following.

$$\frac{\Gamma \vdash A <_c A' : Type \quad \Gamma \vdash B : (A')Type}{\Gamma \vdash \Sigma(A, B \circ c) <_{d_1} \Sigma(A', B) : Type} \quad (size(A) = size(A'))$$

or

$$\frac{\Gamma \vdash A <_c A' : Type \quad \Gamma \vdash B : (A')Type}{\Gamma \vdash \Sigma(A, B \circ c) <_{d_1} \Sigma(A', B) : Type} \quad (size(A) \not> size(A'))$$

In $T[\mathcal{R}\pi_1]_0$, $size$ is well-defined. Similarly, $size$ can be defined in $T[\mathcal{R}]_0$ and one can prove its well-definedness (see [Luo99,LL01] for more details of $T[\mathcal{R}]_0$ and $T[\mathcal{R}]$. Here, $\mathcal{R}$ includes one of the above rules). It is obvious that $T[\mathcal{R}\pi_1]_0$ and $T[\mathcal{R}]_0$ are equivalent in terms of the following lemma.

[3] Thanks to an anonymous referee for the comments on this issue.

Lemma 9. *If $\Gamma \vdash A \propto_c B : Type$ is derivable in $T[\mathcal{R}\pi_1]_0$ then $\Gamma \vdash A <_c B : Type$ is derivable in $T[\mathcal{R}]_0$ and vice versa.*

However, since the system $T[\mathcal{R}]$ includes the Coercive definition rule and the Coercive application rules in Appendix, A and A' in the side-condition may not be well-typed in the original type theory any more. The way to compute such terms is to insert coercions first and then do usual computation in the original type theory. So the property that inserting coercion is decidable in $T[\mathcal{R}]$ must be proved first in order to argue the well-definedness of $size$. There is a circularity, that is, a property of $T[\mathcal{R}]$ is needed in order to present $T[\mathcal{R}]$ itself.

Algorithm and decidability Since we proved the coherence and admissibility of substitution and transitivity, the coercion searching for whole system is decidable if it is decidable for $\mathcal{C}$. In other words, there is an algorithm to check whether there exists a coercion between any two types. We omit the details here.

Further study In this paper, we had a case study about how to combine incoherent coercions. The methods developed here may have a wider application. In general, it is also natural to consider new subtyping relations to block those derivations which make the coercive subtyping system incoherent. The method to introduce new transitivity rules may guide a further study of a system in which there are more than one subtyping relations.

The subtyping rules for parameterised inductive types need further study. For example, we introduce subtyping rules for lists as follows.

$$\frac{\Gamma \vdash A \propto_c B : Type}{\Gamma \vdash List(A) <_d List(B) : Type}$$

where $d = map(A, B, c)$ such that $d(nil(A)) = nil(B)$ and $d(cons(A, a, l)) = cons(B, c(a), d(l))$.

As studied in [LLS02], if we add this rule in the system, the transitivity rules would not be admissible. In a forthcoming paper, we will study new computation rules for parameterised inductive types and such rules will make, for example, $map(B, C, c') \circ map(A, B, c)$ and $map(A, C, c' \circ c)$ computationally equal. And hence the above subtyping rules for lists enjoy the property of admissibility of transitivity.

Related work The early development of the framework of coercive subtyping is closely related to Aczel's idea in type-checking overloading methods for classes [Acz94] and the work on giving coercion semantics to lambda calculi with subtyping by Breazu-Tannen et al [BCGS91]. Barthe and his colleagues have studied constructor subtyping and its possible applications in proof systems [BF99,BvR00]. A recent logical study of subtyping in system **F** can be found in [LMS95] and Chen has studied the issue of transitivity elimination in that framework [Che98].

Acknowlegements We would like to thank the member of the Computer-Assisted Reasoning Group at Durham for discussions and the TYPES03 referees for the comments on the paper.

References

[Acz94] P. Aczel. Simple overloading for type theories. Draft, 1994.

[B+00] B. Barras et al. *The Coq Proof Assistant Reference Manual (Version 6.3.1)*. INRIA-Rocquencourt, 2000.

[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.

[BF99] G. Barthe and M.J. Frade. Constructor subtyping. *Proceedings of ESOP'99, LNCS 1576*, 1999.

[BvR00] G. Barthe and F. van Raamsdonk. Constructor subtyping in the calculus of inductive constructions. *Proceedings of FOSSACS'00, LNCS 1784*, 2000.

[Che98] G. Chen. *Subtyping, Type Conversion and Transitivity Elimination*. PhD thesis, University of Paris VII, 1998.

[CL01] P. Callaghan and Z. Luo. An implementation of LF with coercive subtyping and universes. *Journal of Automated Reasoning*, 27(1):3–27, 2001.

[CLP01] P. C. Callaghan, Z. Luo, and J. Pang. Object languages in a type-theoretic meta-framework. *Workshop of Proof Transformation and Presentation and Proof Complexities (PTP'01)*, 2001.

[HHP87] R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Proc. 2nd Ann. Symp. on Logic in Computer Science. IEEE*, 1987.

[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.

[LC98] Z. Luo and P. Callaghan. Coercive subtyping and lexical semantics (extended abstract). *LACL'98*, 1998.

[LL01] Y. Luo and Z. Luo. Coherence and transitivity in coercive subtyping. In R. Nieuwenhuis and A. Voronkov, editors, *8th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 2250 of *LNAI*, pages 249–265. Springer-Verlag, 2001.

[LLS02] Y. Luo, Z. Luo, and S. Soloviev. Weak transitivity in coercive subtyping. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs*, volume 2646 of *LNCS*, pages 220–239. Springer-Verlag, 2002.

[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.

[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.

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

[SL02] S. Soloviev and Z. Luo. Coercion completion and conservativity in coercive subtyping. Annals of Pure and Applied Logic, 2002.

Appendix: The following are the inference rules for the coercive subkinding extension $T[\mathcal{R}\pi_1]$ (not including the rules for subtyping)

Basic subkinding rule

$$\frac{\Gamma \vdash A <_c B : Type}{\Gamma \vdash El(A) <_c El(B)} \qquad \frac{\Gamma \vdash A \prec_c B : Type}{\Gamma \vdash El(A) <_c El(B)}$$

Coercive application rules

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

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

Coercive definition rule

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

Subkinding for dependent product kinds

$$\frac{\Gamma \vdash K_1' = K_1 \;\; \Gamma, x' : K_1' \vdash K_2 <_c K_2' \;\; \Gamma, x : K_1 \vdash K_2 \;\; kind}{\Gamma \vdash (x:K_1)K_2 <_{[f:(x:K_1)K_2][x':K_1']c(f(x'))} (x':K_1')K_2'}$$

$$\frac{\Gamma \vdash K_1' <_c K_1 \;\; \Gamma, x' : K_1' \vdash [c(x')/x]K_2 = K_2' \;\; \Gamma, x : K_1 \vdash K_2 \;\; kind}{\Gamma \vdash (x:K_1)K_2 <_{[f:(x:K_1)K_2][x':K_1']f(c(x'))} (x':K_1')K_2'}$$

$$\frac{\Gamma \vdash K_1' <_{c_1} K_1 \;\; \Gamma, x' : K_1' \vdash [c_1(x')/x]K_2 <_{c_2} K_2' \;\; \Gamma, x : K_1 \vdash K_2 \;\; kind}{\Gamma \vdash (x:K_1)K_2 <_{[f:(x:K_1)K_2][x':K_1']c_2(f(c_1(x')))} (x':K_1')K_2'}$$

Congruence rules for subkinding

$$\frac{\Gamma \vdash K_1 <_c K_2 \;\; \Gamma \vdash K_1 = K_1' \;\; \Gamma \vdash K_2 = K_2' \;\; \Gamma \vdash c = c' : (K)K'}{\Gamma \vdash K_1' <_{c'} K_2'}$$

Transitivity and Substitution rules for subkinding

$$\frac{\Gamma \vdash K <_c K' \;\; \Gamma \vdash K' <_{c'} K''}{\Gamma \vdash K <_{c' \circ c} K''} \qquad \frac{\Gamma, x:K, \Gamma' \vdash K_1 <_c K_2 \;\; \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]K_1 <_{[k/x]c} [k/x]K_2}$$