Coherence and transitivity in coercive subtyping

Yong Luo and Zhaohui Luo*

Department of Computer Science,
University of Durham,
South Road, Durham, DH1 3LE , U.K.
E-mail: {Yong.Luo, Zhaohui.Luo}@durham.ac.uk

Abstract Coercive subtyping is a general approach to subtyping, inheritance and abbreviation in dependent type theories. A vital requirement for coercive subtyping is that of coherence – computational uniqueness of coercions between any two types. In this paper, we develop techniques useful in proving coherence and its related result on admissibility of transitivity and substitution. In particular, we consider suitable subtyping rules for Π-types and Σ-types and prove its coherence and the admissibility of substitution and transitivity rules at the type level in the coercive subtyping framework.

1 Introduction

Coercive subtyping, as studied in [Luo97,Luo99,SL01], represents a novel general approach to subtyping and inheritance in type theory. In particular, it provides a framework in which subtyping, inheritance, and abbreviation can be understood in dependent type theories where types are understood as consisting of canonical objects.

In this paper, we consider the issue of *coherence* in the framework of coercive subtyping; in particular, we develop techniques useful for proving coherence in coercive subtyping. The coherence conditions are the most basic requirement for the subtyping rules. In essence, it says that any two coercions between two types must be the same, which ensures the uniqueness of coercions (if any). Among other things, coherence is the basis for the whole coercive subtyping framework to be consistent and for it to be implemented in a correct way. A related important issue is that of admissibility of transitivity and substitution, which apart from its relationship with coherence, is essential for implementation of the theory.

We shall develop methods to prove coherence and the admissibility results. In particular, we consider suitable subtyping rules for Π and Σ-types as examples to demonstrate these proof techniques. Although some important meta-theoretic results such as the conservativity result have been obtained for coercive subtyping, the current paper is the first attempt to prove coherence and admissibility

* The first author thanks the support of the ORS Award and the Durham University studentship. This work by the second author is partly supported by the UK EPSRC grant GR/M75518 and the EU grant on the TYPES project.

results at the type level in the framework. The techniques developed here have wider and further applications.

In Section 2, we give an overview of coercive subtyping, presenting the formal framework and giving informal explanations of the coherence problem. In Section 3, a general strategy for proving coherence and the admissibility results is considered, and a formal definition of the so-called well-defined coercions is given as the basis for the proof techniques to be considered in the following sections. In Sections 4 and 5, we prove coherence and the admissibility of substitution and transitivity rules, respectively. Discussions are given in the concluding section (Section 6), where we discuss issues such as decidability and weak transitivity, the latter of which is important for the coercive subtyping framework and its applications.

2 Coercive subtyping and the coherence problem

In this section, we give a brief introduction to coercive subtyping, explain the notion of coherence and its importance, and lay down the necessary formal details to be used in the following sections.

2.1 Coercive subtyping

The basic idea of coercive subtyping, as studied in e.g., [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,Bai99,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,SL01]. 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. The main result of [SL01] 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. (However, how to prove coherence and admissibility of transitivity at the type level has not been studied; this is the subject of this paper.)

Coercion mechanisms with certain restrictions have been implemented both in the proof development system Lego [LP92] and Coq [B+00], by Bailey [Bai99] and Saibi [Sai97], respectively. Callaghan of the Computer Assisted Reasoning Group at Durham has implemented Plastic [CL01], a proof assistant that supports logical framework and coercive subtyping with a mixture of simple coercions, parameterised coercions, coercion rules for parameterised inductive types, and dependent coercions [LS99].

A formal presentation Here, before discussing further the problems of coherence and transitivity, we first give a formal presentation of the framework of coercive cubtyping, which is also the basis for our development in latter sections. We shall be brief in this paper (for details and more explanations, see [Luo99]).

Coercive subtyping is formally formulated as an extension of (type theories specified in) the logical framework LF [Luo94], whose rules are given in Appendix A. In LF, $Type$ represents the conceptual universe of types and $(x:K)K'$ 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 $f(k)$. LF can be used to specify type theories, such as Martin-Löf's type theory [NPS90] and UTT [Luo94].

For example, Π-types, types of dependent functions, can be specified by introducing the constants for (1) formation: $\Pi(A,B)$ is a type for any type A and any family of types B, (2) introduction: $\lambda(A,B,f)$ is a function of type $\Pi(A,B)$ if f is a functional operation of kind $(x:A)B(x)$, and (3) elimination, from which we can define the application operator $app(A,B,F,a)$. Similarly, we can introduce Σ-types $\Sigma(A,B)$ with introduction operator to form $pair(A,B,a,b)$ and an elimination operator from which the projections $\pi_1(A,B,p)$ and $\pi_2(A,B,p)$ can be defined.

Notation We shall use the following notations:

- We shall often omit the El-operator in LF to write A for $El(A)$ when no confusion may occur and may write $(K)K'$ for $(x:K)K'$ when x does not occur free in K'.
- We sometimes use $M[x]$ to indicate that variable x may occur free in M and subsequently write $M[N]$ for $[N/x]M$, when no confusion may occur.
- Functional composition: for $f:(K_1)K_2$ and $g:(y:K_2)K_3[y]$, define $g\circ f =_{\mathrm{df}} [x:K_1]g(f(x)):(x:K_1)K_3[f(x)]$, where x does not occur free in f or g.
- 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 $\vdash K_1 = K_1'$, ..., $x_1:K_1,...,x_{n-1}:K_{n-1} \vdash K_n = K_n'$.

A system with coercive subtyping, $T[\mathcal{R}]$, is an extension of any type theory T specified in LF by a set of basic subtyping rules $\mathcal{R}$. It can be presented in two stages: first we formulate the intermediate system $T[\mathcal{R}]_0$ with subtyping judgements of the form $\Gamma \vdash A <_c B\!:\!Type$, and then add the subkinding judgements of the form $\Gamma \vdash K <_c K'$ and rules concerning coercions between kinds.

$T[\mathcal{R}]_0$ is an extension of T (only) with the subtyping judgement form $\Gamma \vdash A <_c B\!:\!Type$ and 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

$$(Cong)\quad \frac{\begin{array}{l}\Gamma \vdash A <_c B\!:\!Type \\ \Gamma \vdash A = A'\!:\!Type \quad \Gamma \vdash B = B'\!:\!Type \quad \Gamma \vdash c = c'\!:\!(A)B\end{array}}{\Gamma \vdash A' <_{c'} B'\!:\!Type}$$

In the presentation of coercive subtyping in [Luo99], $T[\mathcal{R}]_0$ also has the following substitution and transitivity rules:

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

$$(Trans)\qquad \frac{\Gamma \vdash A <_c B\!:\!Type \quad \Gamma \vdash B <_{c'} C\!:\!Type}{\Gamma \vdash A <_{c' \circ c} C\!:\!Type}$$

Since we consider in this paper how to prove that the substitution and transitivity rules are admissible, we do not include them as basic rules.

Remark 1. We have the following remarks.

- $T[\mathcal{R}]_0$ is obviously a conservative extension of the original type theory T, since the subtyping judgements do not contribute to any derivation of a judgement of any other form.
- The set of basic coercion rules is supposed to be *coherent*; we shall give definition and discussions of this in the next subsection.

The system $T[\mathcal{R}]$, the extension of T with coercive subtyping with respect to $\mathcal{R}$, is the system obtained from $T[\mathcal{R}]_0$ by adding the new subkinding judgement form $\Gamma \vdash K <_c K'$ and the rules in Appendix B. Note that the substitution rule and the transitivity rule for kinds (the last two rules in Appendix B) can be eliminated under the assumption that the set of basic subtyping rules $\mathcal{R}$ is coherent [SL01].

Notation Since we are not much concerned with the subkinding judgements and are mainly concerned with the subtyping judgements, we shall simply write $\Gamma \vdash A <_c B$ for $\Gamma \vdash A <_c B\!:\!Type$, when there is no confusion may occur. Sometimes, we shall also write $\Gamma \vdash A = B$ for $\Gamma \vdash A = B\!:\!Type$.

2.2 Coherence of the basic subtyping rules

The basic subtyping rules are the basis for the coercive subtyping system. Examples of such rules include

- simple coercion declarations such as those between basic inductive types: $Even$ is a subtype of Nat;
- parameterised coercions representing (point-wise) subtyping (or subfamily relation) between two families of types indexed by objects of the same type; 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 type: e.g., $\Sigma(A, B)$ is a subtype of $\Sigma(A', B')$ if A is a subtype of A' and B is a subfamily of B'.

The most basic requirement for such basic subtyping rules is that of coherence, given in the following definition, which essentially says that basic coercions between any two types must be unique.

Definition 1 (coherence condition). *We say that the basic subtyping rules are* coherent *if* $T[\mathcal{R}]_0$ *has the following* coherence properties*:*

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

Remark 2. This is a weaker notion of coherence as compared with that given in [Luo99], since there the rules $(Subst)(Trans)$ are included in $\mathrm{T}[\mathcal{R}]_0$. In general, when parameterised coercions and substitutions are present, coherence is undecidable. This is one of the reasons one needs to consider proofs of coherence in general.

Examples of basic coercion rules include those mentioned above, among which one can find the lifting operators between type universes, overloading coercions, etc. Also, for example, for parameterised Π-types and Σ-types, we can have their subtyping rules as given in Figure 1 and Figure 2. Note that these rules are suitable ones for which we can show that transitivity is admissible. If one chose inductively defined coercions, strong transitivity would not be admissible (see Section 6.2 for discussions.)

3 Well-defined coercions

As mentioned above, unless the coercions can be represented as a finite graph, coherence is in general undecidable, especially when we have parameterised coercions. So we need to consider how to prove coherence and the related admissibility results.

A general strategy we adopt is to consider such proofs in a stepwise way. That is, if we know that some existing coercions (possibly generated by some

Domain rule

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

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

Codomain rule

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

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

Domain-Codomain rule

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

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

Figure1. Basic subtyping rules for Π-types.

First Component rule

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

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

Second Component rule

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

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

First-Second Component rule

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

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

Figure2. Basic subtyping rules for Σ-types.

existing rules) are coherent and have good admissibility properties, and we add some more subtyping rules, can we show that the newly extended system is still coherent and has good admissibility properties? This has led us to define the following concept of *well-defined coercions*. We shall then use subtyping rules for Π and Σ-types to demonstrate how coherence etc can be proved.

Definition 2 (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) $\Gamma \vdash A <_c B \in \mathcal{C}$ implies $\Gamma \vdash A : Type$, $\Gamma \vdash B : Type$ and $\Gamma \vdash c : (A)B$.
   (b) $\Gamma \vdash A <_c A \notin \mathcal{C}$ for any Γ , A , and c.
   (c) $\Gamma \vdash A <_{c_1} B \in \mathcal{C}$ and $\Gamma \vdash A <_{c_2} B \in \mathcal{C}$ imply $\Gamma \vdash c_1 = c_2 : (A)B$.
2. *(Congruence) $\Gamma \vdash A <_c B \in \mathcal{C}$, $\Gamma \vdash A = A'$, $\Gamma \vdash B = B'$ and $\Gamma \vdash c = c' : (A)B$ imply $\Gamma \vdash A' <_{c'} B' \in \mathcal{C}$.*
3. *(Transitivity) $\Gamma \vdash A <_{c_1} B \in \mathcal{C}$ and $\Gamma \vdash B <_{c_2} A' \in \mathcal{C}$ imply $\Gamma \vdash A <_{c_3} A' \in \mathcal{C}$ for some c_3 such that $\Gamma \vdash c_3 = c_2 \circ c_1 : (A)A'$.*
4. *(Substitution) $\Gamma, x : K, \Gamma' \vdash A <_c B \in \mathcal{C}$ implies for any k such that $\Gamma \vdash k : K$, $\Gamma, [k/x]\Gamma' \vdash [k/x]A <_{c'} [k/x]B \in \mathcal{C}$ for some c' such that $\Gamma, [k/x]\Gamma' \vdash c' = [k/x]c : ([k/x]A)[k/x]B$.*
5. *(Weakening) $\Gamma \vdash A <_c B \in \mathcal{C}$, $\Gamma \subseteq \Gamma'$ and Γ' is valid imply $\Gamma' \vdash A <_c B \in \mathcal{C}$.*

Remark 3. A WDC can be thought of as a set of coercions generated from some basic coercions, some basic subtyping rules, and the rules $(Cong)(Subst)(Trans)$ and that of weakening.

We have the following properties of WDCs.

Lemma 1.

1. *If $\Gamma \vdash A <_{c_1} B \in \mathcal{C}$, $\Gamma \vdash B' <_{c_2} A' \in \mathcal{C}$ and $\Gamma \vdash B = B'$, then $\Gamma \vdash A <_{c_3} A' \in \mathcal{C}$ for some c_3 and $\Gamma \vdash c_3 = c_2 \circ c_1 : (A)A'$.*
2. *If $\Gamma, x : K, \Gamma' \vdash A <_c B \in \mathcal{C}$, $\Gamma \vdash K = K'$, then $\Gamma, x : K', \Gamma' \vdash A <_c B \in \mathcal{C}$.*
3. *If $\Gamma \vdash A <_c B \in \mathcal{C}$, $\vdash \Gamma = \Gamma'$, then $\Gamma' \vdash A <_c B \in \mathcal{C}$.*
4. *If $\Gamma \vdash A <_c B \in \mathcal{C}$, $\Gamma' \vdash A' <_{c'} B' \in \mathcal{C}$, $\vdash \Gamma = \Gamma'$, $\Gamma \vdash A = A'$ and $\Gamma \vdash B = B'$, then $\Gamma \vdash c = c' : (A)B$.*

In the following sections, we shall consider the system of coercive subtyping whose basic subtyping rules ($\mathcal{R}$) consist of the following rule, where $\mathcal{C}$ is a WDC:

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

and the Π and Σ-subtyping rules in Figures 1 and 2. Furthermore, we assume that for any judgement $\Gamma \vdash A <_c B \in \mathcal{C}$, neither A nor B is computationally equal to a Π-type or a Σ-type. We denote the derivable subtyping judgements of this system by $\mathcal{C}_{\mathcal{M}}$. We also assume that the original type theory T has good properties, in particular the Church-Rosser property and the property of context replacement by equal kinds. In the following two sections, we shall show that $\mathcal{C}_{\mathcal{M}}$ is also a WDC.

Remark 4. The above system is equivalent to $\mathrm{T}[\mathcal{R}]_0$ where $\mathcal{R}$ consists of (C) and the Π/Σ subtyping rules.

4 Coherence

In this section, we give a proof of coherence of basic subtyping rules of Π-types and Σ-types.

Lemma 2. *If $\Gamma \vdash M_1 <_d M_2 \in \mathcal{C}_{\mathcal{M}}$, then one of the following holds:*

- $\Gamma \vdash M_1 <_d M_2 \in \mathcal{C}$;
- *Both M_1 and M_2 are computationally equal to Π-types; or*
- *Both M_1 and M_2 are computationally equal to Σ-types.*

Proof. By induction on derivations. If $\Gamma \vdash M_1 <_d M_2 \notin \mathcal{C}$, its derivation must end with a Π-subtyping rule, a Σ-subtyping rule, or the congruence rule. If it is one of the Π or Σ-subtyping rules, then we know both M_1 and M_2 are Π-types or Σ-types. If the last rule is the congruence rule $(Cong)$,

$$\frac{\Gamma \vdash M_1' <_{d'} M_2' : Type \quad \Gamma \vdash M_1 = M_1' : Type \quad \Gamma \vdash M_2 = M_2' : Type \quad \Gamma \vdash d' = d : (M_1')M_2'}{\Gamma \vdash M_1 <_d M_2}$$

then by induction hypothesis, the lemma holds for $\Gamma \vdash M_1' <_{d'} M_2'$. If both M_1' and M_2' are computationally equal to Π-types or Σ-types, so are M_1 and M_2. If $\Gamma \vdash M_1' <_{d'} M_2' \in \mathcal{C}$, then $\Gamma \vdash M_1 <_d M_2 \in \mathcal{C}$ because $\mathcal{C}$ is a WDC, which is closed under congruence. $\square$

Lemma 3.

1. *If $\Gamma \vdash \Pi(A,B) <_d \Pi(A',B') \in \mathcal{C}_{\mathcal{M}}$ then $\Gamma \vdash A = A'$ or $\Gamma \vdash A' <_c A \in \mathcal{C}_{\mathcal{M}}$ for some c.*
2. *If $\Gamma \vdash \Sigma(A,B) <_d \Sigma(A',B') \in \mathcal{C}_{\mathcal{M}}$ then $\Gamma \vdash A = A'$ or $\Gamma \vdash A <_c A' \in \mathcal{C}_{\mathcal{M}}$ for some c.*
3. *If $\Gamma \vdash \Pi(A,B) <_d \Pi(A',B') \in \mathcal{C}_{\mathcal{M}}$ and $\Gamma \vdash A = A'$ then $\Gamma, x : A \vdash B(x) <_{e[x]} B'(x) \in \mathcal{C}_{\mathcal{M}}$ for some e.*
4. *If $\Gamma \vdash \Sigma(A,B) <_d \Sigma(A',B') \in \mathcal{C}_{\mathcal{M}}$ and $\Gamma \vdash A = A'$ then $\Gamma, x : A \vdash B(x) <_{e[x]} B'(x) \in \mathcal{C}_{\mathcal{M}}$ for some e.*
5. *If $\Gamma \vdash \Pi(A,B) <_d \Pi(A',B') \in \mathcal{C}_{\mathcal{M}}$ and $\Gamma \vdash A' <_c A \in \mathcal{C}_{\mathcal{M}}$ then $\Gamma, x : A' \vdash B(c(x)) = B'(x)$ or $\Gamma, x : A' \vdash B(c(x)) <_{e[x]} B'(x) \in \mathcal{C}_{\mathcal{M}}$ for some e.*
6. *If $\Gamma \vdash \Sigma(A,B) <_d \Sigma(A',B') \in \mathcal{C}_{\mathcal{M}}$ and $\Gamma \vdash A <_c A' \in \mathcal{C}_{\mathcal{M}}$ then $\Gamma, x : A \vdash B(x) = B'(c(x))$ or $\Gamma, x : A \vdash B(x) <_{e[x]} B'(c(x)) \in \mathcal{C}_{\mathcal{M}}$ for some e.*

Proof. By induction on derivations. We only consider the first statement; the proofs of the others are similar. For the first, a derivation of the judgement $\Gamma \vdash \Pi(A, B) <_d \Pi(A', B')$ must be of the form

$$\dfrac{\dfrac{\dfrac{\vdots}{one\ of\ three\ \Pi - subtyping\ rules}}{\Gamma \vdash \Pi(A_1, B_1) <_{d'} \Pi(A_2, B_2)} \\ \vdots \\ ...(Congruence\ rules)... \\ \vdots}{\Gamma \vdash \Pi(A, B) <_d \Pi(A', B')}$$

where $\Gamma \vdash \Pi(A_1, B_1) = \Pi(A, B){:}\,Type$, $\Gamma \vdash \Pi(A_2, B_2) = \Pi(A', B'){:}\,Type$, and $\Gamma \vdash d' = d{:}\,(C)C'$ for some C and C' computationally equal to $\Pi(A, B)$ and $\Pi(A', B')$, respectively. Hence, by the Church-Rosser theorem of the original type theory T and conservativity of $\mathrm{T}[\mathcal{R}]_0$ over T, $\Gamma \vdash A_1 = A$, $\Gamma \vdash B_1 = B$, $\Gamma \vdash A_2 = A'$ and $\Gamma \vdash B_2 = B'$. So $\Gamma \vdash A = A'$ or $\Gamma \vdash A' <_c A$ by the congruence rule. $\square$

Lemma 4. *If $\Gamma \vdash M_1 <_c M_2 \in \mathcal{C}_{\mathcal{M}}$, then $\Gamma \not\vdash M_1 = M_2$.*

Theorem 1 (Coherence). *If $\Gamma \vdash M_1 <_d M_2 \in \mathcal{C}_{\mathcal{M}}$, $\Gamma' \vdash M_1' <_{d'} M_2' \in \mathcal{C}_{\mathcal{M}}$, $\vdash \Gamma = \Gamma'$, $\Gamma \vdash M_1 = M_1'$, and $\Gamma \vdash M_2 = M_2'$ then $\Gamma \vdash d = d'{:}\,(M_1)M_2$.*

Proof. By induction on derivations. By Lemma 2, we only have to consider the following cases.

- $\Gamma \vdash M_1 <_d M_2 \in \mathcal{C}$. Then, none of M_1 and M_2 is computationally equal to a Π-type or Σ-type. Therefore, nor is M_1' or M_2'. So, by Lemma 2, $\Gamma' \vdash M_1' <_{d'} M_2' \in \mathcal{C}$. Now, by Lemma 1(2), we have $\Gamma \vdash d = d'{:}\,(M_1)M_2$.
- Both M_1 and M_2 are computationally equal to Π-types. Then any derivation of $\Gamma \vdash M_1 <_d M_2$ contains a subderivation whose last rule is one of the Π-subtyping rules followed by congruence rules. We only consider the case the Π-subtyping rule concerned is the third rule in Figure 1; i.e., the derivation is of the form

$$\dfrac{\dfrac{\begin{matrix}\vdots & & \vdots \\ \Gamma \vdash A_2 <_c A_1 & & \Gamma, x : A_2 \vdash B_1(c(x)) <_{e[x]} B_2(x)\end{matrix}}{\Gamma \vdash \Pi(A_1, B_1) <_{d_1} \Pi(A_2, B_2)} \\ \vdots \\ ...(Congruence\ rules)... \\ \vdots}{\Gamma \vdash M_1 <_d M_2}$$

  where $\Gamma \vdash \Pi(A_1, B_1) = M_1$, $\Gamma \vdash \Pi(A_2, B_2) = M_2$, and $\Gamma \vdash d_1 = d{:}\,(M_1)M_2$.

Now, it must be the case that any derivation of $\Gamma' \vdash M_1' <_{d'} M_2'$ must contain a subderivation whose last rule is also the same Π-subtyping rule as above, followed by applications of the congruence rule; i.e., it must be of the form

$$\dfrac{\dfrac{\begin{array}{cc}\vdots & \vdots \\ \Gamma' \vdash A_2' <_{c'} A_1' & \Gamma', x : A_2' \vdash B_1'(c'(x)) <_{e'[x]} B_2'(x)\end{array}}{\Gamma' \vdash \Pi(A_1', B_1') <_{d_1'} \Pi(A_2', B_2')} \quad \begin{array}{c}\cdot \\ ...(Congruence\ rules)... \\ \cdot\end{array}}{\Gamma' \vdash M_1' <_{d'} M_2'}$$

where $\Gamma' \vdash \Pi(A_1', B_1') = M_1'$, $\Gamma' \vdash \Pi(A_2', B_2') = M_2'$, and $\Gamma' \vdash d' = d_1' : (M_1')M_2'$. To see this is the case, by Lemma 3, we only have to show that

1. $\Gamma' \not\vdash A_2' = A_1'$, and
2. $\Gamma', x : A_2' \not\vdash B_1'(c'(x)) = B_2'(x)$.

For the first case, since $\Gamma \vdash M_1 = M_1'$ and $\Gamma \vdash M_2 = M_2'$, we have $\Gamma \vdash \Pi(A_1, B_1) = \Pi(A_1', B_1')$ and $\Gamma \vdash \Pi(A_2, B_2) = \Pi(A_2', B_2')$. Hence, by Church-Rosser theorem in T and conservativity of T$[\mathcal{R}]_0$ over T, $\Gamma \vdash A_1 = A_1'$, $\Gamma \vdash B_1 = B_1'$, $\Gamma \vdash A_2 = A_2'$ and $\Gamma \vdash B_2 = B_2'$. As $\Gamma \vdash A_2 <_c A_1$, we have by Lemma 4, $\Gamma \not\vdash A_2 = A_1$. So $\Gamma' \not\vdash A_2' = A_1'$.

For the second case, a similar argument suffices, except that we use the fact that, by the argument of the first case and induction hypothesis, $\Gamma \vdash c = c' : (A_2)A_1$.

Since the derivations must be of the above forms, by induction hypothesis, we have $\Gamma \vdash c = c' : (A_2)A_1$ and $\Gamma, x : A_2 \vdash e[x] = e'[x] : (B_1(c(x)))B_2(x)$. Hence $\Gamma \vdash d = d' : (M_1)M_2$.

- Both M_1 and M_2 are computationally equal to Σ-types. The proof of this case is similar to the above case. □

5 Admissibility of Substitution and Transitivity

In the presentation of coercive subtyping in [Luo99], substitution and transitivity are two of the basic rules in the theoretical framework. However, in an implementation of coercive subtyping, these rules are ignored simply because that they cannot be directly implemented. For this reason among others, proving admissibility of such rules (or their elimination) is always an important task for any subtyping systems.

In this paper, we do not take substitution and transitivity as basic rules, but we prove that they are both admissible when we extend a WDC by the Π and Σ-subtyping rules. In order to prove admissibility of transitivity, we also need to prove the theorem about weakening.

Theorem 2 (Substitution and weakening).

1. *(Substitution) If* $\Gamma, x : K, \Gamma' \vdash M_1 <_d M_2 \in \mathcal{C}_{\mathcal{M}}$ *and* $\Gamma \vdash k : K$, *then* $\Gamma, [k/x]\Gamma' \vdash [k/x]M_1 <_{d'} [k/x]M_2 \in \mathcal{C}_{\mathcal{M}}$ *for some* d' *such that* $\Gamma, [k/x]\Gamma' \vdash d' = [k/x]d\colon ([k/x]M_1)[k/x]M_2$.
2. *(Weakening) If* $\Gamma \vdash M_1 <_d M_2 \in \mathcal{C}_{\mathcal{M}}$, $\Gamma \subseteq \Gamma'$ *and* Γ' *is valid then* $\Gamma' \vdash M_1 <_d M_2 \in \mathcal{C}_{\mathcal{M}}$.

Proof. By induction on derivations and using Lemma 2. □

To prove the admissibility of transitivity, the usual measures (e.g., the size of types concerned) do not seem to work (or even to be definable), since types essentially involve computations. We use a measure developed by Chen in his PhD thesis [Che98], which only considers subtyping judgements in a derivation, defined as follows.

Definition 3 (depth). *Let* D *be a derivation of a subtyping judgement of the form* $\Gamma \vdash A <_c B\colon Type$. *Then*

1. *If the last rule of* D *is*

$$\frac{\Gamma \vdash A <_c B : Type \in \mathcal{C}}{\Gamma \vdash A <_c B\colon Type}$$

   then $depth(D) = 1$.
2. *If the last rule of* D *is*

$$\frac{S_1,\ \ldots,\ S_n,\quad D_1,\ \ldots,\ D_m}{\Gamma \vdash A <_c B\colon Type}$$

   where $S_1, \ldots, S_n$ *are derivations of subtyping judgements of the form* $\Gamma' \vdash A' <_{c'} B'\colon Type$ *and* $D_1, \ldots, D_m$ *are derivations of other forms of judgements, then* $depth(D) = max\{depth(S_1), \ldots, depth(S_n)\} + 1$.

The following lemma shows 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 5.

1. *If* $\vdash \Gamma = \Gamma'$, $\Gamma \vdash M_1 <_d M_2\colon Type \in \mathcal{C}_{\mathcal{M}}$, *and* D *is a derivation of* $\Gamma \vdash M_1 <_d M_2\colon Type$, *then*
   (a) $\Gamma' \vdash M_1 <_d M_2\colon Type \in \mathcal{C}_{\mathcal{M}}$, *and*
   (b) *there is a derivation* D' *of* $\Gamma' \vdash M_1 <_d M_2\colon Type$ *such that* $depth(D) = depth(D')$.
2. *If* $\Gamma, x : El(A), \Gamma' \vdash M_1 <_{c_1} M_2\colon Type \in \mathcal{C}_{\mathcal{M}}$, $\Gamma \vdash A' <_{c_2} A : Type \in \mathcal{C}_{\mathcal{M}}$, *and* D *is a derivation of* $\Gamma, x : El(A), \Gamma' \vdash M_1 <_{c_1} M_2\colon Type$, *then*
   (a) $\Gamma, y : El(A'), [c_2(y)/x]\Gamma' \vdash [c_2(y)/x]M_1 <_{c_3} [c_2(y)/x]M_2\colon Type \in \mathcal{C}_{\mathcal{M}}$ *for some* c_3 *such that* $\Gamma, y : El(A'), [c_2(y)/x]\Gamma' \vdash c_3 = [c_2(y)/x]c_1\colon ([c_2(y)/x]M_1)[c_2(y)/x]M_2$, *and*

(b) *there is a derivation* D' *of* $\Gamma, y : El(A'), [c_2(y)/x]\Gamma' \vdash [c_2(y)/x]M_1 <_{c_3} [c_2(y)/x]M_2 : Type$ *such that* $depth(D) = depth(D')$.

Proof. By induction on derivations. The key point is that, in the proofs briefly described below, the size of a derivation may change, but the depth of a derivation, which only counts the subtyping judgements, does not.

1. For (1), in the base case, we use Lemma 1(3), and in the step cases, the theorem of context replacement by equal kinds in T and conservativity of $T[\mathcal{R}]_0$ over T.
2. For (2), in the base case, we use the fact that, if $\Gamma, x : El(A), \Gamma' \vdash M_1 <_{c_1} M_2 : Type \in \mathcal{C}$, then $\Gamma, y : El(A'), [c_2(y)/x]\Gamma' \vdash [c_2(y)/x]M_1 <_{c_3} [c_2(y)/x]M_2 : Type \in \mathcal{C}$ for some c_3 such that $\Gamma, y : El(A'), [c_2(y)/x]\Gamma' \vdash c_3 = [c_2(y)/x]c_1 : ([c_2(y)/x]M_1)[c_2(y)/x]M_2$. In the step cases, use of induction hypothesis suffices. □

Now, we can prove the admissibility of transitivity.

Theorem 3 (Transitivity). *If* $\Gamma \vdash M_1 <_{d_1} M_2 \in \mathcal{C}_{\mathcal{M}}$, $\Gamma \vdash M_2' <_{d_2} M_3 \in \mathcal{C}_{\mathcal{M}}$ *and* $\Gamma \vdash M_2 = M_2'$, *then* $\Gamma \vdash M_1 <_{d_3} M_3 \in \mathcal{C}_{\mathcal{M}}$ *for some* d_3 *such that* $\Gamma \vdash d_3 = d_2 \circ d_1 : (M_1)M_3$.

Proof. By induction on $depth(D) + depth(D')$, where D and D' are derivations of $\Gamma \vdash M_1 <_{d_1} M_2$ and $\Gamma \vdash M_2' <_{d_2} M_3$, respectively. In the base case, we have that the judgements $\Gamma \vdash M_1 <_{d_1} M_2$ and $\Gamma \vdash M_2' <_{d_2} M_3$ are both in $\mathcal{C}$. By Lemma 1(1), we have $\Gamma \vdash M_1 <_{d_3} M_3 \in \mathcal{C}$ for some d_3 such that $\Gamma \vdash d_3 = d_2 \circ d_1 : (M_1)M_3$.

In the step case, if $\Gamma \vdash M_1 <_{d_1} M_2$ and $\Gamma \vdash M_2' <_{d_2} M_3$ are both in $\mathcal{C}$, then a similar argument as the base case suffices. Otherwise, we have that either $\Gamma \vdash M_1 <_{d_1} M_2$ or $\Gamma \vdash M_2' <_{d_2} M_3$ is not in $\mathcal{C}$. Therefore, by Lemma 2 and the assumption that $\Gamma \vdash M_2 = M_2'$, all of M_1, M_2, M_2' and M_3 are computationally equal to Π-types or Σ-types. We only consider the case that they are equal to Π-types. Suppose that the derivation D and D' be of the following forms (we only consider the only more difficult example among the combinations of Π-subtyping rules):

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\vdots\\ D_1\\ \vdots\\ \Gamma \vdash A_2 <_{c_1} A_1\end{matrix} \quad \begin{matrix}\vdots\\ D_2\\ \vdots\\ \Gamma, x : A_2 \vdash B_1(c_1(x)) <_{e_1[x]} B_2(x)\end{matrix}}{\Gamma \vdash \Pi(A_1, B_1) <_{d_1'} \Pi(A_2, B_2)}}{\begin{matrix}\vdots\\ ...(Congruence\ rules)...\\ \vdots\end{matrix}}}{\Gamma \vdash M_1 <_{d_1} M_2}$$

where $\Gamma \vdash \Pi(A_1, B_1) = M_1$, $\Gamma \vdash \Pi(A_2, B_2) = M_2$, $\Gamma \vdash d_1' = d_1 : (M_1)M_2$ and $d_1' = [f : \Pi(A_1, B_1)]\lambda(A_2, B_2, [x : A_2]e_1[x](app(A_1, B_1, f, c_1(x))))$, and

$$\dfrac{\dfrac{\begin{array}{c}\vdots \\ D_2' \\ \vdots \\ \Gamma \vdash A_3 <_{c_2} A_2'\end{array} \quad \begin{array}{c}\vdots \\ D_2' \\ \vdots \\ \Gamma, x : A_3 \vdash B_2'(c_2(x)) <_{e_2[x]} B_3(x)\end{array}}{\begin{array}{c}\Gamma \vdash \Pi(A_2', B_2') <_{d_2'} \Pi(A_3, B_3) \\ \vdots \\ ...(Congruence\ rules)... \\ \vdots \end{array}}}{\Gamma \vdash M_2' <_{d_2} M_3}$$

where $\Gamma \vdash \Pi(A_2', B_2') = M_2'$, $\Gamma \vdash \Pi(A_3, B_3) = M_3$, $\Gamma \vdash d_2' = d_2 : (M_2')M_3$ and $d_2' = [f : \Pi(A_2', B_2')]\lambda(A_3, B_3, [x : A_3]e_2[x](app(A_2', B_2', f, c_2(x))))$. We obviously have $depth(D_1) < depth(D)$, $depth(D_2) < depth(D)$, $depth(D_1') < depth(D')$, and $depth(D_2') < depth(D')$.

Now, since $\Gamma \vdash M_2 = M_2'$, we have by Church-Rosser theorem of T and conservativity of $\mathrm{T}[\mathcal{R}]_0$ over T, $\Gamma \vdash A_2 = A_2'$ and $\Gamma \vdash B_2 = B_2' : (A_2)Type$. From the former, $\Gamma \vdash A_3 <_{c_2} A_2$ by the congruence rule. By Lemma 5(2), $\Gamma, x : A_3 \vdash B_1(c_1(c_2(x))) <_{e_3[x]} B_2(c_2(x))$ for some e_3 such that $\Gamma, x : A_3 \vdash e_3[x] = e_1[c_2(x)]$ and there is a derivation D_3 of the judgement $\Gamma, x : A_3 \vdash B_1(c_1(c_2(x))) <_{e_3[x]} B_2(c_2(x))$ and $depth(D_3) = depth(D_2) < depth(D)$.

Now, we have

$$depth(D_1) + depth(D_1') < depth(D) + depth(D')$$

$$depth(D_3) + depth(D_2') < depth(D) + depth(D')$$

By induction hypothesis, there is c_3 such that $\Gamma \vdash A_3 <_{c_3} A_1 \in \mathcal{C}_{\mathcal{M}}$ and $\Gamma \vdash c_3 = c_1 \circ c_2 : (A_3)A_1$. And because $\Gamma, x : A_3 \vdash B_2(c_2(x)) = B_2'(c_2(x))$ (as we have $\Gamma \vdash B_2 = B_2' : (A_2)Type$), by induction hypothesis, there is e_4 such that

$$\Gamma, x : A_3 \vdash B_1(c_1(c_2(x))) <_{e_4[x]} B_3(x) \in \mathcal{C}_{\mathcal{M}}$$

$$\Gamma, x : A_3 \vdash e_4[x] = e_2[x] \circ e_3[x] : (B_1(c_1(c_2(x))))B_3(x).$$

Hence $\Gamma, x : A_3 \vdash e_4[x] = e_2[x] \circ e_1[c_2(x)] : (B_1(c_1(c_2(x))))B_3(x)$. So by the Domain-Codomain rule (the third rule in Figure 1), $\Gamma \vdash \Pi(A_1, B_1) <_{d_3} \Pi(A_3, B_3) \in \mathcal{C}_{\mathcal{M}}$, where

$$d_3 =_{df} [f : \Pi(A_1, B_1)]\lambda(A_3, B_3, [x : A_2]e_4[x](app(A_1, B_1, f, c_3(x))))$$

and we have $d_3 = d_2 \circ d_1$. Finally, by the congruence rule, we have $\Gamma \vdash M_1 <_{d_3} M_3 \in \mathcal{C}_{\mathcal{M}}$. □

Corollary 1. *$\mathcal{C}_{\mathcal{M}}$ is a WDC.*

Proof. By Lemma 4 and Theorems 1, 2, and 3. □

6 Discussions

In this section, we briefly discuss several issues of interest such as those concerning decidability and transitivity, and related work.

6.1 Decidability

Once we have proven coherence and admissibility of substitution and transitivity (as we have done for Π and Σ-subtyping rules), we can be sure that coercion searching is decidable for $\mathcal{C}_{\mathcal{M}}$ if it is decidable for $\mathcal{C}$; in other words, it is decidable whether $\Gamma \vdash A <_c B : Type$ is derivable. One can give a sound and complete algorithm to do this. We omit the details here. This is of course important in implementations.

6.2 Weak transitivity

The transitivity rule $(Trans)$ states that $c' \circ c$ is a coercion from A to C if c and c' are coercions from A to B and from B to C, respectively. In fact, this transitivity rule is very strong. For instance, if we introduce subtyping rule for lists:

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

where d is defined inductively such that $d(nil(A)) = nil(B)$ and $d(cons(A, a, l)) = cons(B, c(a), d(l))$, then the rule $(Trans)$ fails to be admissible.

A weaker version is

$$(WTrans) \qquad \frac{\Gamma \vdash A < B \quad \Gamma \vdash B < C}{\Gamma \vdash A < C}$$

where the judgement $\Gamma \vdash A < B$ means that '$\Gamma \vdash A <_c B$ for some c'. In fact, this weaker version of transitivity seems to be better suited to the wider applications. Furthermore, if the type theory T has a propositional equality $=_A$ (e.g., Leibniz's equality or Martin-Löf's equality type), we can prove that

- If $\Gamma \vdash A <_c B$, $\Gamma \vdash B <_d C$, and $\Gamma \vdash A <_e C$, then e is extensionally equal to $d \circ c$ in the sense that the proposition $\forall x : A.e(x) =_C d(c(x))$ is provable in the type theory.

The admissibility of weak transitivity and the above extensional justification will be discussed in a forthcoming paper [LLS01]. And the admissibility of $(Trans)$ rule and $(WTrans)$ rule in extensional type theory needs futhur study.

6.3 Related work

Besides those mentioned above, the related work includes previous meta-theoretic studies about coercive subtyping. One of the future tasks to be done is to consider how the conservativity result and related work at the kind level [SL01] can be related to the current development and hence to obtain an overall better understanding of the framework. We should mention again Chen's work [Che98], in particular his development of the depth measure, which seems to be very useful in proving admissibility of transitivity.

Acknowledgements We would like to thank Jianming Pang and Sergei Soloviev for their helpful comments and corrections on an earlier draft of this paper, and the members of the Computer-Assisted Reasoning Group at Durham for discussions of the issues concerned. Thanks also go to the LPAR'01 referees who made helpful comments on the paper.

References

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

[Bai99] A. Bailey. *The Machine-checked Literate Formalisation of Algebra in Type Theory.* PhD thesis, University of Manchester, 1999.

[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.

[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.

[LLS01] Y. Luo, Z. Luo, and S. Soloviev. Weak transitivity in coercive subtyping. In preparation, 2001.

[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*, 29, 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.

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

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

[SL01] S. Soloviev and Z. Luo. Coercion completion and conservativity in coercive subtyping. To be published in Annals of Pure and Applied Logic, 2001.

Appendix A

The following gives the rules of the logical framework LF.

Contexts and assumptions

$$\frac{}{\langle\rangle\ valid} \qquad \frac{\Gamma \vdash K\ kind \quad x \notin FV(\Gamma)}{\Gamma, x : K\ valid} \qquad \frac{\Gamma, x : K, \Gamma'\ valid}{\Gamma, x : K, \Gamma' \vdash x : K}$$

Equality rules

$$\frac{\Gamma \vdash K\ kind}{\Gamma \vdash K = K} \qquad \frac{\Gamma \vdash K = K'}{\Gamma \vdash K' = K} \qquad \frac{\Gamma \vdash K = K' \quad \Gamma \vdash K' = K''}{\Gamma \vdash K = K''}$$

$$\frac{\Gamma \vdash k : K}{\Gamma \vdash k = k : K} \qquad \frac{\Gamma \vdash k = k' : K}{\Gamma \vdash k' = k : K} \qquad \frac{\Gamma \vdash k = k' : K \quad \Gamma \vdash k' = k'' : K}{\Gamma \vdash k = k'' : K}$$

$$\frac{\Gamma \vdash k : K \quad \Gamma \vdash K = K'}{\Gamma \vdash k : K'} \qquad \frac{\Gamma \vdash k = k' : K \quad \Gamma \vdash K = K'}{\Gamma \vdash k = k' : K'}$$

Substitution rules

$$\frac{\Gamma, x : K, \Gamma'\ valid \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma'\ valid}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash K'\ kind \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]K'\ kind} \qquad \frac{\Gamma, x : K, \Gamma' \vdash K'\ kind \quad \Gamma \vdash k = k' : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]K' = [k'/x]K'}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash k' : K' \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]k' : [k/x]K'} \qquad \frac{\Gamma, x : K, \Gamma' \vdash k' : K' \quad \Gamma \vdash k_1 = k_2 : K}{\Gamma, [k_1/x]\Gamma' \vdash [k_1/x]k' = [k_2/x]k' : [k_1/x]K'}$$

$$\frac{\Gamma, x : K, \Gamma' \vdash K' = K'' \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]K' = [k/x]K''} \qquad \frac{\Gamma, x : K, \Gamma' \vdash k' = k'' : K' \quad \Gamma \vdash k : K}{\Gamma, [k/x]\Gamma' \vdash [k/x]k' = [k/x]k'' : [k/x]K'}$$

The kind Type

$$\frac{\Gamma\ valid}{\Gamma \vdash Type\ kind} \qquad \frac{\Gamma \vdash A : Type}{\Gamma \vdash El(A)\ kind} \qquad \frac{\Gamma \vdash A = B : Type}{\Gamma \vdash El(A) = El(B)}$$

Dependent product kinds

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

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

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

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

Appendix B

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

New rules for application

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

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

Coercive definition rule

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

Basic subkinding rule

$$\frac{\Gamma \vdash A <_c B\colon Type}{\Gamma \vdash El(A) <_c El(B)}$$

Subkinding for dependent product kinds

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

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

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

Congruence rule for subkinding

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

Transitivity and substitution rules for subkinding

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