On Subtyping in Type Theories with Canonical Objects

Georgiana Elena Lungu
Department of Computer Science, Royal Holloway, University of London, U.K.
georgiana.lungu.2013@live.rhul.ac.uk

Zhaohui Luo[1]
Department of Computer Science, Royal Holloway, University of London, U.K.
Zhaohui.Luo@rhul.ac.uk

Abstract

How should one introduce subtyping into type theories with canonical objects such as Martin-Löf's type theory? It is known that the usual subsumptive subtyping is inadequate and it is understood, at least theoretically, that coercive subtyping should instead be employed. However, it has not been studied what the proper coercive subtyping mechanism is and how it should be used to capture intuitive notions of subtyping. In this paper, we introduce a type system with signatures where coercive subtyping relations can be specified, and argue that this provides a suitable subtyping mechanism for type theories with canonical objects. In particular, we show that the subtyping extension is well-behaved by relating it to the previous formulation of coercive subtyping. The paper then proceeds to study the connection with intuitive notions of subtyping. It first shows how a subsumptive subtyping system can be embedded faithfully. Then, it studies how Russell-style universe inclusions can be understood as coercions in our system. And finally, we study constructor subtyping as an example to illustrate that, sometimes, injectivity of coercions need be assumed in order to capture properly some notions of subtyping.

2012 ACM Subject Classification F.3.1 Logics and Meanings of Programs

Keywords and phrases subtyping, type theory, conservative extension, canonical objects

Digital Object Identifier 10.4230/LIPIcs.TYPES.2016.13

1 Introduction

Type theories with canonical objects such as Martin-Löf's type theory [26] have been used as the basis for both theoretical projects such as Homotopy Type Theory [32] and practical applications in proof assistants such as Coq [10] and Agda [1]. In this paper, we investigate how to extend such type theories with subtyping relations, an issue that is important both theoretically and practically, but has not been settled.

Subsumptive Subtyping. The usual way to introduce subtyping is via the following subsumption rule:

$$\frac{a : A \quad A \leq B}{a : B}$$

This is directly related to the notion of subset in mathematics and naturally linked to type assignment systems in programming languages like ML or Haskell. However, subsumptive subtyping is not adequate for type theories with canonical objects since it would destroy key

[1] This work is partially supported by the EU COST Action CA15123 and the CAS/SAFEA International Partnership Program.

properties of such type theories including canonicity (every object of an inductive type is equal to a canonical object) and subject reduction (computation preserves typing) [21, 16].

For instance, the Russell-style type universes $U_i : U_{i+1}$ $(i \in \omega)$ [23] constitute a special case of subsumptive subtyping with $U_i \leq U_{i+1}$ [18]. If we adopt the standard notation of terms with full type information, the resulting type theory with Russell-style universes would fail to have canonicity or subject reduction.[2] An alternative is to use proof terms with less typing information like using (a, b) instead of $pair(A, B, a, b)$ to represent pairs, as in HoTT (see Appendix 2 of [32]). The problem with this approach is that not only the property of type uniqueness fails, but a proof term may have incompatible types. For example, for $a : A$ and $A : U$, where U is a type universe, the pair (A, a) has both types $U \times A$ and $\Sigma X{:}U.X$, which are incompatible in the sense that none of them is a subtype of the other. This would lead to undecidability of type checking,[3] which is unacceptable for type theories with logics based on the propositions-as-types principle.

In §3 we will show how we can embed a subtyping system with the above subsumption rule into the coercive subtyping system we introduce in this paper.

Coercive Subtyping. An alternative way to introduce subtyping is coercive subtyping, where a subtyping relationship between two types is modelled by means of a unique coercion between them. The early developments of coercion semantics of subtyping for programming languages include [25, 29, 28, 6], among others. At the theoretical level, previous work on coercive subtyping for dependent type theories such as [15, 21] show that coercive subtyping can be adequately employed for dependent type theories with canonical objects to preserve the meta-theoretic properties such as canonicity and normalisation of the original type theories. Based on this, coercive subtyping has been successfully used in various applications based on the implementations of coercions in Coq and several other proof assistants [30, 3, 7].

However, the theoretical research on coercive subtyping such as [21] considers a rather abstract way of extension with coercive subtyping. For any type theory T, it extends it with a (coherent, but possibly infinite) set C of subtyping judgements to form a new type theory $T[C]$. Although this is well-suited in a theoretical study, it does not tell one how the extension should be formulated concretely in practice. In fact, a proposal of adding coercive subtyping assumptions in contexts [22] has met with potential difficulties in meta-theoretic studies that cast doubts on the seemingly attractive proposal. The complication was caused by the fact that coercion relations specified in a context can be moved to the right of the turnstile sign $\vdash$ to introduce terms with the so-called local coercions that are only effective in a localised scope. It is still unknown whether such mechanisms can be employed successfully. This has partly led to the current research that studies a more restrictive calculus that only allows coercive subtyping relations to be specified in signatures whose entries cannot be localised in terms.

Main Contributions. In this paper, we study a type theory with signatures where coercive subtyping relations can be specified and argue that this provides a suitable subtyping mechanism for type theories with canonical objects.[4] This claim is backed up by first showing

[2] See §4.1 of the current paper for an example of the former and §4.3 of [16] for an example of the latter.

[3] To see the problem of type checking, it may be worth pointing out that, for a dependent type theory, type checking depends on type inference; put in another way, in a type-checking algorithm one has to infer the type of a term in many situations.

[4] A type theory with signatures was also proposed by the second author in [19] in the context of applying type theories to natural language semantics.

that the subtyping extension is conservative over the original type theory and that all its valid derivations correspond to valid derivations in the original calculus, and then studying its connection with subsumptive subtyping and its use in modelling some of the intuitive notions of subtyping including that induced by Russell-style universes in type theory.

The notion of signature in type theory was first studied in the Edinburgh Logical Framework [12] with judgements of the form $\Gamma \vdash_\Sigma J$, where the signatures Σ are used to describe constants of a logical system, in contrast with the contexts Γ that introduce variables which can be abstracted to the right of the turnstile sign by means of quantification or λ-abstraction. We will introduce the notion of signature by extending (the typed version of) Martin-Löf's logical framework LF (Chapter 9 of [14]) to obtain the system LF_S, which can be used similarly as LF in specifying type theories such as Martin-Löf's type theory [26]. Formulating the coercive subtyping relation in a type theory based on a logical framework makes it possible to extend the formulation to other type constructors too. We then introduce Π_S, a system with Π-types specified in LF_S, and $\Pi_{S,\leq}$ that extends Π_S to allow specification in signatures of subtyping entries $A \leq_c B$ that specifies that A is a subtype of B via coercion c, a function from A to B. We will justify that the coercive subtyping mechanism is abbreviational by showing that $\Pi_{S,\leq}$ is equivalent to a similar system as previously studied [21] and hence has desirable properties [31, 13, 33].

Although it is incompatible with the notion of canonical objects, subsumptive subtyping is widely used and, intuitively, it is the concept in mind in the first place when considering subtyping. It is therefore worth studying its relationship with the coercive subtyping calculus. Aspinall and Compagnoni [2] approached the topic of subsumptive subtyping in dependent type theory by developing a type system, with contextual subsumptive subtyping entries of the form $\alpha \leq A$ to declare that the type variable α is a subtype of A, and its checking algorithm in the Edinburgh Logical Framework. In this paper we shall define a subsumptive subtyping system in LF_S, one similar to that in [2], and prove that it can be faithfully embedded in $\Pi_{S,\leq}$.

It is worth noting that subtyping becomes particularly complicated in the case of dependent types. In a type system with contextual subtyping entries such as $\alpha \leq A$ as in Aspinall and Compagnoni's system, one has to decide whether to allow abstraction (for example, by λ or Π) over the subtyping entries. If one did, it would lead to types with bounded quantification of the form $\Pi\alpha \leq A.B$, which would result in complications and, most likely, undecidability of type checking (cf., Pierce's work that shows undecidability of type checking in $F_{\leq}$, an extension of the second-order λ-calculus with subtyping and bounded quantification [27]). In order to avoid bounded quantification, Aspinall and Compagnoni [2] present the subtyping entries in contexts, but do not enable their moving to the right of $\vdash$. In consequence, abstraction by λ or Π of those entries that occur to the left of a subtyping entry is obstructed. We chose to represent subtyping entries in the signatures in order to allow abstraction to happen freely for contextual entries.

We shall then consider two case studies, showing how coercive subtyping may be used to capture an intuitive notion of subtyping. Type universes [23] are our first example here. The Russell-style universes constitute a typical example of subsumptive subtyping. The second author [18] observed that, although subsumptive subtyping causes problems with the notion of canonicity, one can obtain the essence of Russell-style universes by means of Tarski-style universes together with coercive subtyping by taking the explicit lifting operators between Tarski-style universes as coercions. Our embedding theorem (Theorem 34) that relates subsumptive and coercive subtyping can be extended for type systems with universes, therefore justifying this claim.

Subsumptive subtyping, esp. in its extreme forms, intuitively embodies a notion of injectivity that is in general not the case for coercive subtyping. One of such extreme forms of subtyping is constructor subtyping [4]. As the second case study, we shall relate it to our coercive subtyping system and show that, once equipped with injectivity of coercions, coercive subtyping can faithfully model the notion of injectivity intuitively assumed in subsumptive subtyping.

Related Work. Subtyping has been studied extensively both for type systems of programming languages and type theories implemented in proof assistants. Early studies of subtyping for programming languages have considered both subsumptive and coercive subtyping, mainly for simpler and non-dependent type systems (see, for example, [25, 29, 28, 6]). For example, Reynolds [28] considered extrinsic and intrinsic models of coercions and their applications to programming.

Subtyping in dependent type theories has been studied by Aspinall and Compagnoni [2] for Edinburgh LF, Betarte and Tasistro [5] about subkinding between kinds (called types) for Martin-Löf's logical framework, and Barthe and Frade [4] on constructor subtyping, among others. A theoretical framework of coercive subtyping for type theories with canonical objects has been developed and studied by the second author and colleagues in a series of papers and PhD theses [15, 21, 31, 13, 33]. In this setting, any dependent type theory T can be extended with coercive subtyping by giving a (possibly infinite) set C of basic subtyping judgements, resulting in the extended calculus $T[C]$. The meta-theory of such a calculus $T[C]$ was first studied in [31] where, among other things, the basic approach to proving that coercive subtyping is an abbreviational extension was developed, which was further studied and improved in, for example, [13, 33]. Coercions have been implemented in several proof assistants such as Coq [10, 30], Lego [20, 3], Matita [24] and Plastic [7] and used effectively for large proof development and, more recently, in formal development of natural language semantics based on type theory [17, 8, 9].

The above framework of coercive subtyping [21] has served as a theoretical tool to show in principle that coercive subtyping is adequate for type theories with canonical objects. However, as pointed out above, such a theoretical framework does not serve as a concrete system in practice. In this paper, we shall use subtyping entries in signatures to specify basic subtyping relations and study the resulting calculus, both in meta-theory and in practical modelling.

In §2, we present $\Pi_{S,\leq}$ and study its meta-theoretic properties. §3 presents a subsumptive subtyping system based on [2] and shows that it can be embedded faithfully in $\Pi_{S,\leq}$. The two case studies on type universes and injectivity are studied in §4, with the relationship between Russell-style and Tarski-style universes studied in §4.1 and constructor subtyping and injectivity in §4.2. The Conclusion discusses possible further research directions.

2 Coercive Subtyping in Signatures

We aim to introduce a calculus that can model intuitive notions of subtyping such as subsumption and, at the same time, preserves the desirable properties of the original type theory. In this section, we present $\Pi_{S,\leq}$, a type system with signatures where we can specify coercive subtyping relations, and then study its properties by relating it to the earlier formulation of coercive subtyping.

In what follows we use $\equiv$ for syntactic identity and assume that the signatures are coherent.

2.1 $\Pi_{S,\leq}$, a Type Theory with Subtyping in Signatures

2.1.1 Logical Framework with Signatures

Type theories can be specified in a logical framework such as Martin-Löf's logical framework [26] or its typed version LF [14]. We shall extend LF with signatures to obtain LF_S.

Informally, a signature is a sequence of entries of several forms, one of which is the form of membership entries $c : K$, which is the traditional form of entries as occurred in contexts (we shall add another form of entries in the next section). If a signature has only membership entries, it is of the form $c_1 : K_1, ..., c_n : K_n$.

▶ Remark (Constants and Variables). Intuitively, we shall call c declared in a signature entry $c : K$ as a constant, while x in a contextual entry $x : K$ as a variable. The formal difference is that, as declared in a signature entry, c cannot be substituted or abstracted (to the right of $\vdash$), while x declared in a contextual entry can either be substituted or abstracted by λ or Π (see later for the formal details.)

LF_S is a dependent type theory whose types are called *kinds* to distinguish them from types in the object type theory. It has the kind $Type$ of all types of the object type theory and dependent Π-kinds of the form $(x{:}K)K'$, which can be written as $(K)K'$ if $x \notin FV(K')$, whose objects are $\lambda-$abstractions of the form $[x{:}K]b$. For each type $A : Type$, we have a kind $El(A)$ which is often written just as A. In LF_S, there are six forms of judgements:

- Σ *valid*, asserting that Σ is a valid signature.
- $\vdash_\Sigma \Gamma$, asserting that Γ is a valid context under Σ.
- $\Gamma \vdash_\Sigma K$ *kind*, asserting that K is a kind in Γ under Σ.
- $\Gamma \vdash_\Sigma k : K$, asserting that k is an object of kind K in Γ under Σ.
- $\Gamma \vdash_\Sigma K_1 = K_2$, asserting that K_1 and K_2 are equal kinds in Γ under Σ.
- $\Gamma \vdash_\Sigma k_1 = k_2 : K$, asserting that k_1 and k_2 are equal objects of kind K in Γ under Σ.

The inference rules of the logical framework LF_S are given in Figure 1; they are the same as those of LF [14], except that we have judgements for signature validity, all other forms of judgements are adjusted accordingly with signatures attached, and we include some structural rules such as those for weakening and signature and context replacement (or signature and contextual equality), as done in the previous formulations in, for example, [21, 31, 33].

2.1.2 Type Theory with Π-types

Let Π_S be the type system with Π-types specified in LF_S. These Π-types are specified in the logical framework by introducing the constants, together with the definition rule, in Figure 2. Note that, with the constants in Figure 2, the rules in Figure 3 become derivable.

2.1.3 Subtyping Entries in Signatures

We present the whole system $\Pi_{S,\leq}$. First, subtyping is represented by means of two forms of judgements:

- subtyping judgements $\Gamma \vdash_\Sigma A \leq_c B : Type$, and
- subkinding judgements $\Gamma \vdash_\Sigma K \leq_c K'$.

Subtyping relations between types (not kinds) can be specified in a signature by means of entries $A \leq_c B : Type$ (or simply written as $A \leq_c B$), where A and B are types and

Validity of Signature/Contexts, Assumptions

$$\frac{}{\langle\rangle\ valid} \qquad \frac{\vdash_\Sigma K\,kind \quad c \notin dom(\Sigma)}{\Sigma, c{:}K\ \ valid} \qquad \frac{\vdash_{\Sigma,c:K,\Sigma'} \Gamma}{\Gamma \vdash_{\Sigma,c:K,\Sigma'} c{:}K}$$

$$\frac{\Sigma\ \ valid}{\vdash_\Sigma \langle\rangle} \qquad \frac{\Gamma \vdash_\Sigma K\,kind \quad x \notin dom(\Sigma) \cup dom(\Gamma)}{\vdash_\Sigma \Gamma, x{:}K} \qquad \frac{\vdash_\Sigma \Gamma, x{:}K, \Gamma'}{\Gamma, x{:}K, \Gamma' \vdash_\Sigma x{:}K}$$

Weakening

$$\frac{\Gamma \vdash_{\Sigma,\ \Sigma'} J \quad \vdash_\Sigma K\ kind \quad c \notin dom(\Sigma, \Sigma')}{\Gamma \vdash_{\Sigma,\ c:K,\ \Sigma'} J} \qquad \frac{\Gamma, \Gamma' \vdash_\Sigma J \quad \Gamma \vdash_\Sigma K\ kind \quad x \notin dom(\Gamma, \Gamma')}{\Gamma, x{:}K, \Gamma' \vdash_\Sigma J}$$

Equality Rules

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

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

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

Signature Replacement

$$\frac{\Gamma \vdash_{\Sigma_0, c:L, \Sigma_1} J \quad \vdash_{\Sigma_0} L = L'}{\Gamma \vdash_{\Sigma_0, c:L', \Sigma_1} J}$$

Context Replacement

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma J \quad \Gamma_0 \vdash_\Sigma K = K'}{\Gamma_0, x{:}K', \Gamma_1 \vdash_\Sigma J}$$

Substitution Rules

$$\frac{\vdash_\Sigma \Gamma_0, x{:}K, \Gamma_1 \quad \Gamma_0 \vdash_\Sigma k{:}K}{\vdash_\Sigma \Gamma_0, [k/x]\Gamma_1}$$

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma K'\ \ kind \quad \Gamma_0 \vdash_\Sigma k{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]K'\ \ kind} \qquad \frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma L = L' \quad \Gamma_0 \vdash_\Sigma k{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]L = [k/x]L'}$$

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma k'{:}K' \quad \Gamma_0 \vdash_\Sigma k{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]k'{:}[k/x]K'} \qquad \frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma l = l'{:}K' \quad \Gamma_0 \vdash_\Sigma k{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]l = [k/x]l'{:}[k/x]K'}$$

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma K'\ \ kind \quad \Gamma_0 \vdash_\Sigma k = k'{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]K' = [k'/x]K'} \qquad \frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma l{:}K' \quad \Gamma_0 \vdash_\Sigma k = k'{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]l = [k'/x]l{:}[k/x]K'}$$

Dependent Product Kinds

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

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

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

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

The kind Type

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

Figure 1 Inference Rules for LF_S

Constant declarations:

$$\begin{array}{rcl} \Pi & : & (A{:}\mathit{Type})(B{:}(A)\mathit{Type})\mathit{Type} \\ \lambda & : & (A{:}\mathit{Type})(B{:}(A)\mathit{Type})((x{:}A)B(x))\Pi(A,B) \\ \mathit{app} & : & (A{:}\mathit{Type})(B{:}(A)\mathit{Type})(\Pi(A,B))(x{:}A)B(x) \end{array}$$

Definitional equality rule

$$\mathit{app}(A,B,\lambda(A,B,f),a) = f(a) : B(a).$$

Figure 2 Constants for Π-types in logical framework

$$\frac{\Gamma \vdash_\Sigma A : \mathit{Type} \quad \Gamma, x{:}A \vdash_\Sigma B(x) : \mathit{Type}}{\Gamma \vdash_\Sigma \Pi(A,B) : \mathit{Type}}$$

$$\frac{\Gamma \vdash_\Sigma A : \mathit{Type} \quad \Gamma \vdash_\Sigma B : (A)\mathit{Type} \quad \Gamma \vdash_\Sigma f : (x{:}A)B(x)}{\Gamma \vdash_\Sigma \lambda(A,B,f) : \Pi(A,B)}$$

$$\frac{\Gamma \vdash_\Sigma g : \Pi(A,B) \quad \Gamma \vdash_\Sigma a : A}{\Gamma \vdash_\Sigma \mathit{app}(A,B,g,a) : B(a)}$$

$$\frac{\begin{array}{c}\Gamma \vdash_\Sigma A : \mathit{Type} \quad \Gamma \vdash_\Sigma B : (A)\mathit{Type} \\ \Gamma \vdash_\Sigma f : (x{:}A)B(x) \quad \Gamma \vdash_\Sigma a : A\end{array}}{\Gamma \vdash_\Sigma \mathit{app}(A,B,\lambda(A,B,f),a) = f(a) : B(a)}$$

Figure 3 Inference Rules for Π_S

$c : (A)B$.[5]

The specifications of subtyping relations are also required to be *coherent*. Coherence is crucial as it ensures a coercive application abbreviates a unique functional application. To define this notion of coherence, we need to introduce a subsystem of $\Pi_{S,\leq}$, called $\Pi^{0K}_{S,\leq}$, defined by the rules of Π_S together with those in Figures 4 and 5, where in the rule for dependent products in Figures 4, the notation $c_2[x]$ was explained in, for example, [16]: it means that x may occur free in c_2, although only inessentially[6]. The composition of functions is defined as follows: For $f{:}(K1)K2$ and $g{:}(K2)K3$, $g \circ f = [x{:}K1]g(f(x)){:}(K1)K3$.

Here is the definition of coherence of a signature, which intuitively says that, under a coherent signature, there cannot be two different coercions between the same types.

▶ **Definition 1.** A signature Σ is **coherent** if, in $\Pi^{0K}_{S,\leq}$, $\Gamma \vdash_\Sigma A \leq_c B$ and $\Gamma \vdash_\Sigma A \leq_{c'} B$ imply $\Gamma \vdash_\Sigma c = c' : (A)B$.

Note that, in comparison with earlier formulations such as [21], we have switched from strict subtyping relation $<$ to $\leq$ and the coherence condition is changed accordingly as well; in particular, under a coherent signature, any coercion from a type to itself must be equal to the identity function. (This is a special case of the above condition when $B \equiv A$: because we

[5] Using some types not contained in $\Pi_{S,\leq}$, more interesting subtyping relations can be specified. For example, for $A \leq_c B$, we could have $A \equiv \mathit{Vect}(N,n)$, $B \equiv \mathit{List}(N)$ and c maps vector $< m_1, ..., m_n >$ to list $[m_1, ..., m_n]$. We shall not formally deal with such extended type systems in the current paper, but the ideas and results are expected to extend to the type systems with such data types (eg, all those in Martin-Löf's type theory).

[6] For instance, one might have (by using the congruence rule) $x{:}A \vdash_\Sigma B \leq_{([y:A]e)(x)} B'$, where $B \leq_e B'$ and $x \notin FV(e)$.

Signature Rules for Subtyping

$$\frac{\vdash_\Sigma A : Type \quad \vdash_\Sigma B : Type \quad \vdash_\Sigma c : (A)B}{\Sigma, A \leq_c B \ \ valid} \qquad \frac{\vdash_{\Sigma_0, A \leq_c B:Type, \Sigma_1} \Gamma}{\Gamma \vdash_{\Sigma_0, A \leq_c B:Type, \Sigma_1} A \leq_c B : Type}$$

Congruence

$$\frac{\Gamma \vdash_\Sigma A \leq_c B : Type \quad \Gamma \vdash_\Sigma A = A' : Type \quad \Gamma \vdash_\Sigma B = B' : Type \quad \Gamma \vdash_\Sigma c = c' : (A)B}{\Gamma \vdash_\Sigma A' \leq_{c'} B' : Type}$$

Transitivity

$$\frac{\Gamma \vdash_\Sigma A \leq_c A' : Type \quad \Gamma \vdash_\Sigma A' \leq_{c'} A'' : Type}{\Gamma \vdash_\Sigma A \leq_{c' \circ c} A'' : Type}$$

Weakening

$$\frac{\Gamma \vdash_{\Sigma,\ \Sigma'} A \leq_d B : Type \quad \vdash_\Sigma K \ kind}{\Gamma \vdash_{\Sigma,\ c:K,\ \Sigma'} A \leq_d B : Type} \quad (c \notin dom(\Sigma, \Sigma'))$$

$$\frac{\Gamma, \Gamma' \vdash_\Sigma A \leq_d B : Type \quad \Gamma \vdash_\Sigma K \ kind}{\Gamma, x{:}K, \Gamma' \vdash_\Sigma A \leq_d B : Type} \quad (x \notin dom(\Gamma, \Gamma'))$$

Signature Replacement

$$\frac{\Gamma \vdash_{\Sigma_0, c:L, \Sigma_1} A \leq_d B : Type \quad \vdash_{\Sigma_0} L = L'}{\Gamma \vdash_{\Sigma_0, c:L', \Sigma_1} A \leq_d B : Type}$$

Context Replacement

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma A \leq_d B : Type \quad \Gamma_0 \vdash_\Sigma K = K'}{\Gamma_0, x{:}K', \Gamma_1 \vdash_\Sigma A \leq_d B : Type}$$

Substitution

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_\Sigma A \leq_c B{:}Type \quad \Gamma_0 \vdash_\Sigma k{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_\Sigma [k/x]A \leq_{[k/x]c} [k/x]B}$$

Identity Coercion

$$\frac{\Gamma \vdash_\Sigma A : Type}{\Gamma \vdash_\Sigma A \leq_{[x:A]x} A : Type}$$

Dependent Product

$$\frac{\Gamma \vdash_\Sigma A' \leq_{c_1} A : Type \quad \Gamma \vdash_\Sigma B, B' : (A)Type \quad \Gamma, x{:}A \vdash_\Sigma B(x) \leq_{c_2[x]} B'(x) : Type}{\Gamma \vdash_\Sigma \Pi(A, B) \leq_d \Pi(A', B' \circ c_1) : Type}$$

where $d \equiv [F : \Pi(A, B)]\lambda(A', B' \circ c_1, [x{:}A']c_2[x](app(A, B, F, c_1(x))))$.

Figure 4 Inference Rules for $\Pi^{0K}_{S,\leq}$ (1)

always have $A \leq_{[x:A]x} A$, if $A \leq_c A$, then $c = [x{:}A]x : (A)A$.) Note also that, it is easy to prove by induction that, if $\Gamma \vdash_\Sigma A \leq_c B : Type$, then $\Gamma \vdash_\Sigma A, B : Type$ and $\Gamma \vdash_\Sigma c : (A)B$.

It is also important to note the difference between a judgement with signature in the current calculus and that in the calculus employed in [21] where there are no signatures. For example, the signatures Σ_1 that contains $A \leq_c B$ and Σ_2 that contains $A \leq_d B$ can both be coherent signatures even when $c \neq d$, while such a situation can only be considered in the earlier setting by having two different type systems $T[C_1]$ and $T[C_2]$, which is rather

Basic Subkinding Rule and Identity Coercion

$$\frac{\Gamma \vdash_{\Sigma} A \leq_c B{:}Type}{\Gamma \vdash_{\Sigma} El(A) \leq_c El(B)} \qquad \frac{\Gamma \vdash_{\Sigma} K \;\; kind}{\Gamma \vdash_{\Sigma} K \leq_{[x:K]x} K}$$

Structural Subkinding Rules

$$\frac{\Gamma \vdash_{\Sigma} K_1 \leq_c K_2 \quad \Gamma \vdash_{\Sigma} K_1 = K_1' \quad \Gamma \vdash_{\Sigma} K_2 = K_2' \quad \Gamma \vdash_{\Sigma} c = c'{:}(K_1)K_2}{\Gamma \vdash_{\Sigma} K_1' \leq_{c'} K_2'}$$

$$\frac{\Gamma \vdash_{\Sigma} K \leq_c K' \quad \Gamma \vdash_{\Sigma} K' \leq_{c'} K''}{\Gamma \vdash_{\Sigma} K \leq_{c' \circ c} K''}$$

$$\frac{\Gamma \vdash_{\Sigma,\ \Sigma'} K \leq_d K' \quad \vdash_{\Sigma} K_0 \; kind}{\Gamma \vdash_{\Sigma,\ c:K_0,\ \Sigma'} K \leq_d K'} \quad (c \notin dom(\Sigma, \Sigma'))$$

$$\frac{\Gamma, \Gamma' \vdash_{\Sigma} K \leq_d K' \quad \Gamma \vdash_{\Sigma} K_0 \; kind}{\Gamma, x{:}K_0, \Gamma' \vdash_{\Sigma} K \leq_d K'} \quad (x \notin dom(\Gamma, \Gamma'))$$

$$\frac{\Gamma \vdash_{\Sigma_0, c:L, \Sigma_1} K \leq_d K' \quad \vdash_{\Sigma_0} L = L'}{\Gamma \vdash_{\Sigma_0, c:L', \Sigma_1} K \leq_d K'} \quad \frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_{\Sigma} L \leq_d L' \quad \Gamma_0 \vdash_{\Sigma} K = K'}{\Gamma_0, x{:}K', \Gamma_1 \vdash_{\Sigma} L \leq_d L'}$$

$$\frac{\Gamma_0, x{:}K, \Gamma_1 \vdash_{\Sigma} K_1 \leq_c K_2 \quad \Gamma_0 \vdash_{\Sigma} k{:}K}{\Gamma_0, [k/x]\Gamma_1 \vdash_{\Sigma} [k/x]K_1 \leq_{[k/x]c} [k/x]K_2}$$

Subkinding for Dependent Product Kind

$$\frac{\Gamma \vdash_{\Sigma} K_1' \leq_{c_1} K_1 \quad \Gamma, x{:}K_1 \vdash_{\Sigma} K_2 \; kind \quad \Gamma, x'{:}K_1' \vdash_{\Sigma} K_2' \; kind \quad \Gamma, x{:}K_1 \vdash_{\Sigma} [c_1(x')/x]K_2 \leq_{c_2} K_2'}{\Gamma \vdash_{\Sigma} (x{:}K_1)K_2 \leq_{[f:(x:K_1)K_2][x':K_1']c_2(f(c_1(x')))} (x{:}K_1')K_2'}$$

Figure 5 Inference Rules for $\Pi^{0K}_{S,\leq}$ (2)

Coercive Application

$$(CA_1) \quad \frac{\Gamma \vdash_\Sigma f{:}(x{:}K)K' \quad \Gamma \vdash_\Sigma k_0{:}K_0 \quad \Gamma \vdash_\Sigma K_0 \leq_c K}{\Gamma \vdash_\Sigma f(k_0){:}[c(k_0)/x]K'}$$

$$(CA_2) \quad \frac{\Gamma \vdash_\Sigma f = f'{:}(x{:}K)K' \quad \Gamma \vdash_\Sigma k_0 = k_0'{:}K_0 \quad \Gamma \vdash_\Sigma K_0 \leq_c K}{\Gamma \vdash_\Sigma f(k_0) = f'(k_0'){:}[c(k_0)/x]K'}$$

Coercive Definition

$$(CD) \quad \frac{\Gamma \vdash_\Sigma f{:}(x{:}K)K' \quad \Gamma \vdash_\Sigma k_0{:}K_0 \quad \Gamma \vdash_\Sigma K_0 \leq_c K}{\Gamma \vdash_\Sigma f(k_0) = f(c(k_0)){:}[c(k_0)/x]K'}$$

Figure 6 The coercive application and definition rules in $\Pi_{S,\leq}$

cumbersome to say the least.[7]

We can, at this point, complete the specification of the system $\Pi_{S,\leq}$ as the extension of $\Pi^{0K}_{S,\leq}$ by adding the rules in Figure 6.

▶ Remark. We can now explain why we have to present the system $\Pi^{0K}_{S,\leq}$ first. The reason is that the coercive definition rule (CD) will force any two coercions to be equal. Therefore, we cannot define the notion of coherence for the system including the (CD) rule as, if we did so, every signature would be coherent by definition.

2.2 Coherence for Kinds and Conservativity

In this subsection, we prove two basic properties of $\Pi_{S,\leq}$: (1) coherence, as defined for types, extends to kinds; (2) it is a conservative extension of the system Π_S.

2.2.1 Coherence for Kinds

Note that the coherence definition refers to types. In what follows we prove that coherence for types implies coherence for kinds. We categorise kinds and show that they can be related via definitional equality or subtyping only if they are of the same category. For this we also define the degree of a kind which intuitively denotes how many dependent product occurrences are in a kind.

▶ **Lemma 2.** *If $\Gamma \vdash_\Sigma A \leq_c B$ is derivable in $\Pi^{0K}_{S,\leq}$ then $\Gamma \vdash_\Sigma c{:}(A)B$ is derivable in $\Pi^{0K}_{S,\leq}$.*

Proof. By induction on the structure of derivations ◀

▶ **Lemma 3.** *If $\Gamma \vdash K \leq_c L$ is derivable in $\Pi^{0K}_{S,\leq}$ then $\Gamma \vdash c{:}(K)L$ is derivable in $\Pi^{0K}_{S,\leq}$.*

Proof. By induction on the structure of derivations. We consider $K \equiv (x{:}K_1)K_2$ and $L \equiv (x{:}L_1)L_2$. If a derivation tree for $\Gamma \vdash K \leq_c L$ ends with the rule for dependent product kind with premises $\Gamma \vdash_\Sigma L_1 \leq_{c_1} K_1, \Gamma, x{:}K_1 \vdash_\Sigma K_2\ kind, \Gamma, y{:}L_1 \vdash_\Sigma L_2\ kind$ and $\Gamma, y{:}L_1 \vdash_\Sigma [c_1(y)/x]K_2 \leq_{c_2} L_2$. By IH we have $\Gamma \vdash_\Sigma c_1{:}(L_1)K_1$ and $\Gamma, y{:}L_1 \vdash_\Sigma c_2{:}([c_1(y)/x]K_2)L_2$. By weakening $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma c_2{:}([c_1(y)/x]K_2)L_2$ and $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma c_1{:}(L_1)K_1$. We have $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma y{:}L_1$ so by application $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma c_1(y){:}K_1$. We have $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma f{:}(x{:}K_1)K_2$ so by application we have $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma$

[7] This has some unexpected consequences concerning parameterised coercions as well. But it is a topic beyond the current paper and will be discussed somewhere else.

$f(c_1(y))$:$[c_1(y)/x]K_2$. By application again we have $\Gamma, f{:}(x{:}K_1)K_2, y{:}L_1 \vdash_\Sigma c_2(f(c_1(y))){:}L_2$ and by abstraction $\Gamma \vdash_\Sigma [f{:}(x{:}K_1)K_2][y{:}L_1]c_2(f(c_1(y))){:}((x{:}K_1)K_2)(y{:}L_1)L_2$ ◀

▶ **Lemma 4.** *Let $\Gamma \vdash_\Sigma K \leq_c L$ be derivable in $\Pi^{0K}_{S,\leq}$. Then K and L are of the same form, i.e., both are El-terms or both are dependent product kinds. Furthermore,*

- *if $K \equiv El(A)$ and $L \equiv El(B)$, then $\Gamma \vdash_\Sigma A \leq_c B : Type$ is derivable in $\Pi^{0K}_{S,\leq}$ and*
- *if $K \equiv (x{:}K_1)K_2$ and $L \equiv (x{:}L_1)L_2$, then $\Gamma \vdash_\Sigma K_1$ kind, $\Gamma, x{:}K_1 \vdash_\Sigma K_2$ kind, $\Gamma \vdash_\Sigma L_1$ kind, and $\Gamma, x{:}L_1 \vdash_\Sigma L_2$ kind are derivable in $\Pi^{0K}_{S,\leq}$.*

The following lemma states that, if there is a subtyping relation between two dependent kinds, then the coercion can be obtained by the subtyping for dependent product kind rule from Figure 5. Note that for this to hold it is essential that we only have subtyping entries in signatures and not subkinding.

▶ **Lemma 5.** *If $\Gamma \vdash_\Sigma (x{:}K_1)K_2 \leq_d (y{:}L_1)L_2$ is derivable in $\Pi^{0K}_{S,\leq}$ then there exist derivable judgements in $\Pi^{0K}_{S,\leq}$, $\Gamma \vdash_\Sigma c_1{:}(L_1)K_1$ and $\Gamma, y{:}L_1 \vdash_\Sigma c_2{:}([c_1(y)/x]K_2)L_2$ s.t.*

- *$\Gamma \vdash_\Sigma L_1 \leq_{c_1} K_1$*
- *$\Gamma, y{:}K_1' \vdash_\Sigma [c_1(y)/x]K_2 \leq_{c_2} L_2$ and*
- *$\Gamma \vdash_\Sigma d = [f{:}(x{:}K_1)K_2][y{:}L_1]c_2(f(c_1(y))){:}((x{:}K_1)K_2)(y{:}L_1)L_2$*

are derivable in $\Pi^{0K}_{S,\leq}$.

Proof. By induction on the structure of derivation of $\Gamma \vdash_\Sigma (x{:}K_1)K_2 \leq_d (y{:}L_1)L_2$. The only non trivial case is when it comes from transitivity.

$$\frac{\Gamma \vdash_\Sigma (x{:}K_1)K_2 \leq_{d_1} C \quad \Gamma \vdash_\Sigma C \leq_{d_2} (y{:}L_1)L_2}{\Gamma \vdash_\Sigma (x{:}K_1)K_2 \leq_{d_2 \circ d_1} (y{:}L_1)L_2}$$

By the previous lemma $\Gamma \vdash_\Sigma C \equiv (z{:}M_1)M_2$. By IH we have that

- $\Gamma \vdash_\Sigma M_1 \leq_{c_1'} K_1$
- $\Gamma, z{:}M_1 \vdash_\Sigma [c_1'(z)/x]K_2 \leq_{c_2'} M_2$
- $\Gamma \vdash_\Sigma d_1 = [f{:}(x{:}K_1)K_2][z{:}M_1]c_2'(f(c_1'(z))){:}((x{:}K_1)K_2)(z{:}M_1)M_2$

and

- $\Gamma \vdash_\Sigma L_1 \leq_{c_1''} M_1$
- $\Gamma, y{:}L_1 \vdash_\Sigma [c_1''(y)/z]M_2 \leq_{c_2''} L_2$
- $\Gamma \vdash_\Sigma d_2 = [f{:}(z{:}M_1)M_2][y{:}L_1]c_2''(f(c_1''(y))){:}((z{:}M_1)M_2)(y{:}L_1)L_2$

are derivable. We apply transitivity to obtain $\Gamma \vdash_\Sigma L_1 \leq_{c_1' \circ c_1''} K_1$ and by weakening and substitution in addition, $\Gamma, y{:}L_1 \vdash_\Sigma [c_1'(c_1''(y))/x]K_2 \leq_{c_2'' \circ [c_1''(y)/z]c_2'} L_2$ and what is left to prove is that $\Gamma \vdash_\Sigma d_2 \circ d_1 = [f{:}(x{:}K_1)K_2][y{:}L_1](c_2'' \circ [c_1''(y)/z]c_2')(f((c_1' \circ c_1'')(y))){:}((x{:}K_1)K_2)(y{:}L_1)L_2$.

Let $\Gamma \vdash_\Sigma F{:}(x{:}K_1)K_2$

$$\begin{aligned}
d_2 \circ d_1(F) &= d_2(d_1(F)) \\
&= d_2([f{:}(x{:}K_1)K_2][z{:}M_1]c_2'(f(c_1'(z)))(F)) \\
&= d_2([F/f][z{:}M_1]c_2'(f(c_1'(z)))) \\
&= d_2([z{:}M_1]c_2'(F(c_1'(z)))) \\
&= ([f{:}(z{:}M_1)M_2][y{:}L_1]c_2''(f(c_1''(y))))([z{:}M_1]c_2'(F(c_1'(z)))) \\
&= [z{:}M_1]c_2'(F(c_1'(z)))/f]([y{:}L_1]c_2''(f(c_1''(y)))) \\
&= [y{:}L_1]c_2''([z{:}M_1]c_2'(F(c_1'(z)))(c_1''(y))) \\
&= [y{:}L_1]c_2''([c_1''(y)/z]c_2'(F(c_1'(c_1''(y))))) \\
&= [y{:}L_1]c_2''([c_1''(y)/z]c_2'(F(c_1'(c_1''(y))))) \\
&= [y{:}L_1](c_2'' \circ [c_1''(y)/z]c_2')(F((c_1' \circ c_1'')(y))) \\
&= ([f{:}(x{:}K_1)K_2][y{:}L_1](c_2'' \circ [c_1''(y)/z]c_2')(f((c_1' \circ c_1'')(y))))(F)
\end{aligned}$$

◀

The following de

nition gives us a measure for the structure of kinds. We will use this measure when proving coherence for kinds. It is particularly important and we will use the fact that this measure is not increased by substitution.

► **Definition 6.** For $\Gamma \vdash_\Sigma K$ we define the **degree of** K where $\Gamma \vdash_\Sigma K$ *kind* as $deg(K) \in \mathbb{N}$ as follows:

1. $deg(Type) = 1$
2. $deg(El(A)) = 1$
3. $deg((x{:}K)L) = deg(K) + deg(L)$

► **Lemma 7.** *The following hold:*

- *if* $\Gamma \vdash_\Sigma K = L$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *then* $deg(K) = deg(L)$
- *if* $\Gamma \vdash_\Sigma K \leq_c L$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *then* $deg(K) = deg(L)$

Proof. We do induction on the structure of derivations of $\Gamma \vdash_\Sigma K = L$ respectively $\Gamma \vdash_\Sigma K \leq L$. For example if it comes from the rule

$$\frac{\Gamma \vdash_\Sigma K_1 = K_2 \quad \Gamma, x{:}K_1 \vdash_\Sigma K_1' = K_2'}{\Gamma \vdash_\Sigma (x{:}K_1)K_1' = (x{:}K_2)K_2'}$$

by IH, $deg(K_1) = deg(K_2)$ and $deg(K_1') = deg(K_2')$, hence $deg((x{:}K_1)K_1') = deg((x{:}K_2)K_2')$

◀

► **Lemma 8** (Coherence for Kinds)**.** *If* $\Gamma \vdash_\Sigma K \leq_c L$ *and* $\Gamma \vdash_\Sigma K \leq_{c'} L$ *are derivable in* $\Pi^{0K}_{S,\leq}$, *then* $\Gamma \vdash_\Sigma c = c' : (K)L$ *is derivable in* $\Pi^{0K}_{S,\leq}$.

Proof. By induction on $n = deg(K)$.

1. For $n = 1$:
   - If $\Gamma \vdash_\Sigma K = El(A)$ and $\Gamma \vdash_\Sigma L = El(B)$ then by Lemma 4 we have $\Gamma \vdash_\Sigma A \leq_c B$ and $\Gamma \vdash_\Sigma A \leq_{c'} B$ and from coherence for types $\Gamma \vdash_\Sigma c = c'{:}(A)B$, hence $\Gamma \vdash_\Sigma c = c'{:}(K)L$
   - If $\Gamma \vdash_\Sigma K = Type$ and $\Gamma \vdash_\Sigma L = Type$ then we can only have $\Gamma \vdash_\Sigma c = Id{:}(K)L$.
2. For $n > 1$, $\Gamma \vdash_\Sigma K \equiv (x{:}K_1)K_2$ and $\Gamma \vdash_\Sigma L \equiv (x{:}L_1)L_2$, by Lemma 5
   - $\Gamma \vdash_\Sigma L_1 \leq_{c_1} K_1$,

- $\Gamma, x{:}K_1 \vdash_\Sigma [c_1(y)/x]K_2 \leq_{c_2} L_2$ and
- $\Gamma \vdash_\Sigma c = [f{:}(x{:}K_1)K_2][y{:}L_1]c_2(f(c_1(y))){:}((x{:}K_1)K_2)(y{:}L_1)L_2$

are derivable for some $\Gamma \vdash_\Sigma c_1{:}(L_1)K_1$ and $\Gamma, x{:}K_1 \vdash_\Sigma c_2{:}([c_1(x)/x]K_2)L_2$ and $deg(L_1)$, $deg(K_1)$, $deg([c_1(y)/x]K_2)$, $deg(L_2)$ are all smaller than n. If

- $\Gamma \vdash_\Sigma L_1 \leq_{c'_1} K_1$,
- $\Gamma, x{:}K_1 \vdash_\Sigma [c'_1(y)/x]K_2 \leq_{c'_2} L_2$ and
- $\Gamma \vdash_\Sigma c' = [f{:}(x{:}K_1)K_2][y{:}L_1]c'_2(f(c'_1(y))){:}((x{:}K_1)K_2)(y{:}L_1)L_2$

are derivable for some other coercions $\Gamma \vdash_\Sigma c'_1{:}(L_1)K_1$ and $\Gamma, x{:}K_1 \vdash_\Sigma c'_2{:}([c'_1(y)/x]K_2)L_2$ then by IH we have $\Gamma \vdash_\Sigma c_1 = c'_1{:}(L_1)K_1$ and $\Gamma, x{:}K_1 \vdash_\Sigma c_2 = c'_2{:}([c'_1(y)/x]K_2)L_2$ and we are done.

◀

2.2.2 Conservativity

Here we prove that, if the signatures are coherent, our calculus $\Pi_{S,\leq}$ is conservative over Π_S in the traditional sense. It follows directly from the fact that $\Pi_{S,\leq}$ keeps track of subtyping entries in the signatures and it carries them along in derivations. More precisely we prove that if a judgement is derivable in $\Pi_{S,\leq}$ and not in Π_S then it cannot be written in Π_S.

The following two lemmas state that any subtyping or subkinding judgement can only be derived with a signature containing subtyping entries, and hence the signature cannot be written in Π_S.

▶ **Lemma 9.** *If $\Gamma \vdash_\Sigma A \leq_c B{:}Type$ is derivable in $\Pi_{S,\leq}$, then Σ contains at least a subtyping entry or $\Gamma \vdash_\Sigma A = B{:}Type$ and $\Gamma \vdash_\Sigma c = Id{:}(A)A$ are derivable in $\Pi_{S,\leq}$.*

Proof. By induction on the structure of derivation. For example if it comes from transitivity from premises $\Gamma \vdash_\Sigma A \leq_c A' : Type$ and $\Gamma \vdash_\Sigma A' \leq_{c'} B : Type$ then the statement simply is true by IH. ◀

▶ **Lemma 10.** *If $\Gamma \vdash_\Sigma K \leq_c L$ is derivable in $\Pi_{S,\leq}$, then Σ contains at least a subtyping entry or $\Gamma \vdash_\Sigma K = L$ and $\Gamma \vdash_\Sigma c = Id{:}(K)L$ are derivable in $\Pi_{S,\leq}$.*

Proof. By induction on the structure of derivation. For example if it comes from transitivity from premises $\Gamma \vdash_\Sigma K \leq_c M$ and $\Gamma \vdash_\Sigma M \leq_{c'} L$ then the statement simply is true by IH.

If it comes from the rule

$$\frac{\Gamma \vdash_\Sigma A \leq_c B{:}Type}{\Gamma \vdash_\Sigma El(A) \leq_c El(B)}$$

then it follows from $\Gamma \vdash_\Sigma A \leq_c B{:}Type$ by the previous lemma ◀

The following lemma extends the statement to express the fact that it is enough for a judgement to contain a non trivial subtyping or subkinding entry (not the identity coercion) in its derivation tree to have a signature that cannot be written in Π_S.

▶ **Lemma 11.** *If D is a valid derivation tree for $\Gamma \vdash_\Sigma J$ in $\Pi_{S,\leq}$ and $\Gamma_1 \vdash_{\Sigma_1} K_1 \leq_{c_0} K_2$ is present in D then, either Σ contains at least a subtyping entry or $\Gamma_1 \vdash_{\Sigma_1} K_1 = K_2$ and $\Gamma_1 \vdash_{\Sigma_1} c_0 = Id_{K_1}{:}(K_1)K_1$ are derivable in $\Pi_{S,\leq}$.*

Proof. If $\Gamma \vdash_\Sigma J$ is a subtyping or subkinding judgement it follows directly from the previous lemmas 9, 5. Likewise, if the judgement comes from a coercive application or coercive definition rule with one of the premises $\Gamma \vdash_\Sigma K \leq L$, then, by the previous lemma the statement holds. Otherwise we do induction on the structure of derivations of $\Gamma \vdash_\Sigma J$. For example if the derivation tree containing the subkinding judgement ends with the rule

$$\frac{\Gamma \vdash_\Sigma K\, kind \quad \Gamma, x{:}K \vdash_\Sigma K'\, kind}{\Gamma \vdash_\Sigma (x{:}K)K' \;\; kind}$$

then the subkinding judgements must be in at least one of the subderivations concluding $\Gamma \vdash_\Sigma K \;\; kind$ and $\Gamma, x{:}K \vdash_\Sigma K' \;\; kind$. The statement then holds by induction hypothesis. ◀

The following lemma states that, if a judgements is derived in $\Pi_{S,\leq}$ using only trivial coercions, then it can be derived in Π_S.

▶ **Lemma 12.** *If in a derivation tree of a judgement derivable in $\Pi_{S,\leq}$ which is not subtyping or subkinding judgement all of the subtyping and subkinding judgements are of the form $\Gamma_1 \vdash_{\Sigma_1} A \leq_{Id_A} A{:}Type$ respectively $\Gamma_1 \vdash_{\Sigma_1} K \leq_{[x:K]x} K$ then the judgement is derivable in Π_S.*

Proof. By induction on the structure of derivations. If the derivation tree D that only contains trivial coercions ends with one of the rules of Π_S,

$$\frac{\frac{D_1}{J_1} \cdots \frac{D_n}{J_n}}{J}(R)$$

then J_1,..., J_n also have derivation trees D_1,...,D_n which only contain at most trivial coercions, hence, by IH, they are derivable in Π_S. We can apply to them, with D_1,...,D_n replaced by their derivation in Π_S the same rule R to obtain the judgement J and the derivation is in Π_S.

Otherwise, if for example the derivation containing only trivial coercions ends with coercive application

$$\frac{\Gamma \vdash_\Sigma f{:}(x{:}K)K' \quad \Gamma \vdash_\Sigma k_0{:}K \quad \Gamma \vdash_\Sigma K \leq_{[x:K]x} K}{\Gamma \vdash_\Sigma f(k_0){:}[[x{:}K]x(k_0)/x]K'}$$

$\Gamma \vdash_\Sigma [[x{:}K]x(k_0)/x]K' = [k_0/x]K'$ and $\Gamma \vdash_\Sigma f{:}(x{:}K)K'$ and $\Gamma \vdash_\Sigma k_0{:}K$ are derivable in Π_S by IH, and from them it follows directly by functional application, in Π_S, $\Gamma \vdash_\Sigma f(k_0){:}[k_0/x]K'$ ◀

▶ **Theorem 13** (Conservativity). *If a judgement is derivable in $\Pi_{S,\leq}$ but not in Π_S, its signature will contain subtyping entries, and hence it cannot be written in Π_S.*

Proof. From the previous lemma, a judgement can only be derivable in $\Pi_{S,\leq}$ but not in Π_S when it contains in all of its derivation trees non trivial subtyping or subkinding judgements. If the judgements is itself a subtyping or subkinding judgement then it vacuously cannot be written in Π_S. Otherwise, by lemma 11 it follows that either all of the subtyping and subkinding judgements are of the form $\Gamma_1 \vdash_{\Sigma_1} A \leq_{Id_A} A{:}Type$ respectively $\Gamma_1 \vdash_{\Sigma_1} K \leq_{[x:K]x} K$ in which case the judgement is derivable in Π_S or its signature contains subtyping entries, in which case it cannot be written in Π_S. ◀

2.3 Justification of $\Pi_{S,\leq}$ as a Well Behaved Extension

We shall show in this subsection that extending the type theory Π_S by coercive subtyping in signatures results in a well-behaved system. In order to do so, we relate the extension with the previous formulation: more precisely, for every signature Σ, we consider a corresponding system $\Pi[C_\Sigma]^{;}$, which is similar to the system $T[C_\Sigma]$ in [21, 33], and we prove the equivalence between judgements in $\Pi_{S,\leq}$ and judgements in such corresponding systems from the point of view of derivability. (see Theorems 22 and 29 below for a more precise description).

This way we argue that there exists a stronger relation between the extension with coercive subtyping entries and the base system based on the fact that was shown in [21, 33] that every derivation tree in $T[\mathcal{C}]$ the extension can be translated to a derivation tree in T such that their conclusion are equal.

2.3.1 The relation between $\Pi^{0K}_{S,\leq}$ and Π_S

Here we show that, if a judgement J is derivable in $\Pi^{0K}_{S,\leq}$, we obtain a set of judgements, one of which is of same as J up to erasing the subtyping entries from a signature. The idea here is that, for any the valid signature in $\Pi^{0K}_{S,\leq}$ and all the judgements using it, we can remove the subtyping entries from it to obtain a valid signature in Π_S and corresponding judgements using this signature.

▶ **Definition 14.** We define $erase(\cdot)$, a map which simply removes subtyping entries from signature as follows:
- $erase(<>) = <>$
- $erase(\Sigma, c{:}K) = erase(\Sigma), c{:}K$
- $erase(\Sigma, A \leq_c B) = erase(\Sigma)$

The following lemma is a completion of weakening and signature replacement for the cases when a signature is weakened with subtyping entries or a subtyping entry is replaced in the signature.

▶ **Lemma 15.**
- *If* $\Gamma \vdash_{\Sigma,\Sigma'} J$ *and* $\vdash_{\Sigma,A\leq_c B:Type,\Sigma'} \Gamma$ *are derivable in* $\Pi^{0K}_{S,\leq}$ *then* $\Gamma \vdash_{\Sigma,A\leq_c B:Type,\Sigma'} J$ *is derivable in* $\Pi^{0K}_{S,\leq}$.
- *If* $\Gamma \vdash_{\Sigma,A\leq_c B\Sigma'} J$, $\vdash_\Sigma A = A'{:}Type$, $\vdash_\Sigma B = B'{:}Type$, $\vdash_\Sigma c = c'{:}(A)B$ $\vdash_{\Sigma,A'\leq_{c'} B':Type,\Sigma'} \Gamma$ *are derivable in* $\Pi^{0K}_{S,\leq}$ *then* $\Gamma \vdash_{\Sigma,A'\leq_{c'} B':Type,\Sigma'} J$ *is derivable in*$\Pi^{0K}_{S,\leq}$.

Proof. By induction on the structure of derivation. ◀

▶ **Lemma 16.** *For* $\Sigma \equiv \Sigma_0, A_0 \leq_{c_0} B_0, \Sigma_1, ..., A_{n-1} \leq_{c_{n-1}} B_{n-1}, \Sigma_n$ *a valid signature as above we will consider the following judgements judgements* $(\star)$ $\vdash_{erase(\Sigma_0,...,\Sigma_i)} c_i{:}(A_i)B_i$, *where* $i \in 0, ..., n$. *Then the following statements hold:*
1. $\vdash_\Sigma \Gamma$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *if and only if* $\vdash_{erase(\Sigma)} \Gamma$ *and* $(\star)$ *are derivable in* Π_S.
2. $\Gamma \vdash_\Sigma J$ *is not a subtyping judgement and is derivable in* $\Pi^{0K}_{S,\leq}$ *if and only if* $\Gamma \vdash_{erase(\Sigma)} J$ *and* $(\star)$ *are derivable in* Π_S.
3. *If* $\Gamma \vdash_\Sigma A \leq_c B$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *then* $\Gamma \vdash_{erase(\Sigma)} c{:}(A)B$ *and* $(\star)$ *are derivable in* Π_S.
4. *If* $\Gamma \vdash_\Sigma K \leq_c L$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *then* $\Gamma \vdash_{erase(\Sigma)} c{:}(K)L$ *and* $(\star)$ *are derivable in* Π_S.

Proof. The only if implication for the first three cases is straightforward by induction on the structure of derivations as subtyping judgements do not contribute to deriving any other type of judgement in $\Pi^{0K}_{S,\leq}$. For the if implication, Lemma 15 is used. The last two points also follow by induction. ◀

2.3.2 $\Pi[\mathcal{C}]^{;}$

Here we consider a system $\Pi[\mathcal{C}]^{;}$ similar to the system $T[\mathcal{C}]$ as presented in [21, 33] with T being the type theory with Π-types.

Here we consider a system similar to the system $T[\mathcal{C}]$ from [21, 33] with dependent product. The difference is that here we fix some prefixes of the context, not allowing substitution and abstraction for these prefixes. In more details, the judgements of $T[\mathcal{C}]^{;}$ will be of the form $\Sigma;\Gamma \vdash J$ instead of $\Gamma \vdash J$, where Σ and Γ are just contexts and substitution and abstraction can be applied to entries in Γ but not Σ. We call this system $\Pi[\mathcal{C}]^{;}$. To delimitate these prefixes we use the symbol ";" and the judgements forms will be as follows:

- $\vdash \Sigma;\Gamma$ signifies a judgement of valid context
- $\Sigma;\Gamma \vdash K\ \ kind$
- $\Sigma;\Gamma \vdash k{:}K$
- $\Sigma;\Gamma \vdash K = K'$
- $\Sigma;\Gamma \vdash k = k'{:}K$

The rules of the system $\Pi[\mathcal{C}]^{;}$ are the ones in Figures 8,9,10, 11 and 12 in the appendix. The difference between these rules and those described in [21, 33] is that, in addition to regular contexts, they also refer to the prefixes apart from substitution and abstraction which is only available for regular contexts. More detailed, we duplicate contexts, assumptions, weakening, context replacement. For all other rules we adjust them to the new forms of judgements by replacing $\Gamma \vdash J$ with $\Sigma;\Gamma \vdash J$. Notice that we do not duplicate substitution as only the context at the righthand side of the ; supports substitution. We will consider the system $\Pi[\mathcal{C}]^{;}_{0K}$ to be the one without coercive application and definition rules, namely the ones in figures 8,9,10 and 11. $\mathcal{C}$ is formed of subtyping judgements and we have the following rule in $\Pi[\mathcal{C}]^{;}_{0K}$

$$\frac{\Gamma \vdash A \leq_c B \in \mathcal{C}}{\Gamma \vdash A \leq_c B}$$

For the system $T[\mathcal{C}]$ coercive application is added as an abbreviation to ordinary functional application and this is ensured by coercive definition together *coherence* of $\mathcal{C}$. Indeed, it was proved in [21, 33] that, when $\mathcal{C}$ is coherent, $\Pi[\mathcal{C}]$ is a well behaved extension of $\Pi[\mathcal{C}]_{0K}$ in that every valid derivation tree D in $\Pi[\mathcal{C}]$ can be translated into a valid derivation tree D' in $\Pi[\mathcal{C}]_{0K}$ and the conclusion of D is definitionally equal to the conclusion of D' in $\Pi[\mathcal{C}]$. We want to avoid doing the complex proof in [21, 33] again and assume that the properties of $\Pi[\mathcal{C}]$ carry over to $\Pi[\mathcal{C}]^{;}$. So next we give the definition of coherence for the set $\mathcal{C}$.

▶ **Definition 17.** The set $\mathcal{C}$ of subtyping judgements is coherent if the following two conditions hold in $\Pi[\mathcal{C}]^{;}_{0K}$:

- If $\Sigma;\Gamma \vdash A \leq_c B$ is derivable, then $\Sigma;\Gamma \vdash c{:}(A)B$ is derivable.
- If $\Sigma;\Gamma \vdash A \leq_c B$ and $\Sigma;\Gamma \vdash A \leq_{c'} B$ are derivable, then $\Sigma;\Gamma \vdash c = c'{:}(A)B$ is derivable.

Notice that in the original formulation $\Sigma;\Gamma \vdash A \leq_{[x:A]x} A$ was not allowed. However the condition that $\Sigma;\Gamma \nvdash A \leq_c A$ was used to prove that a judgement cannot come from both coercive application and functional application. However with the current condition one can prove that, if this is the case, the coercion has to be equal to the identity.

2.3.3 The relation between $\Pi[\mathcal{C}]^{;}$ and $\Pi_{S,\leq}$

Although there is a difference between the new $\Pi_{S,\leq}$ and $\Pi[\mathcal{C}]^{;}$ which lies mainly in the fact that, by introducing coercive subtyping via signature, we introduce them locally to the specific signature, this allowing us to have more coercions between two types under the same kinding assumptions(of the form c:K, x:K) and still have coherence satisfied, whereas by enriching a system with a set of coercive subtyping, our coercions are introduced globally and only one coercion(up to definitional equality) can exist between two types under the same kinding assumptions. However, because signatures are technically just prefix of contexts for which abstraction and substitution are not available [12], we naturally expect that there is a relation between $\Pi_{S,\leq}$ and $\Pi[\mathcal{C}]^{;}$. And indeed here we shall show that for any valid signature Σ in $\Pi_{S,\leq}$, we can represent a class of judgements of $\Pi_{S,\leq}$ depending on Σ as judgements in a $\Pi[\mathcal{C}_\Sigma]^{;}$.

First we consider just $\Pi^{0K}_{S,\leq}$ and $\Pi[\mathcal{C}]^{;}_{0K}$ which are the systems without coercive application and coercive definition and we define a way to transfer coercive subtyping entries of a signature Σ in $\Pi^{0K}_{S,\leq}$ to a set of coercive subtyping judgements of $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$.

▶ **Definition 18.** Let Σ be a signature (not necessarily valid) in $\Pi^{0K}_{S,\leq}$ we define Γ_Σ as follows:

- $\Gamma_{<>} =<>$
- $\Gamma_{\Sigma_0,k:K} = \Gamma_{\Sigma_0}, k:K$
- $\Gamma_{\Sigma_0,A\leq_c B:Type} = \Gamma_{\Sigma_0}$

If Σ is valid in $\Pi^{0K}_{S,\leq}$ we define $\mathcal{C}_\Sigma$ as follows:

- $\mathcal{C}_{<>} = \emptyset$
- $\mathcal{C}_{\Sigma_0,k:K} = \mathcal{C}_{\Sigma_0}$
- $\mathcal{C}_{\Sigma_0,A\leq_c B:Type} = \mathcal{C}_{\Sigma_0} \cup \{\Gamma_{\Sigma_0};<>\vdash A \leq_c B:Type\}$

▶ **Lemma 19.** *If $\Sigma \equiv \Sigma_0, A \leq_c B{:}Type, \Sigma_1$ valid is derivable in $\Pi^{0K}_{S,\leq}$, then $\Gamma_\Sigma \equiv \Gamma_{\Sigma_0,\Sigma_1}$ and $\mathcal{C}_\Sigma = \mathcal{C}_{\Sigma_0,\Sigma_1} \cup \{\Gamma_{\Sigma_0};<>\vdash A \leq_c B{:}Type\}$*

Proof. By induction on the length of Σ. ◀

▶ **Lemma 20.** *Let Σ_1, Σ_3 and $\Sigma_1, \Sigma_2, \Sigma_3$ be valid signatures in $\Pi^{0K}_{S,\leq}$. If J is derivable in $\Pi[\mathcal{C}_{\Sigma_1,\Sigma_3}]^{;}_{0K}$ then J is derivable in $\Pi[\mathcal{C}_{\Sigma_1,\Sigma_2,\Sigma_3}]^{;}_{0K}$*

Proof. By induction on the structure of derivation of J. ◀

First we mention the following notation which we will use throughout the section and which is really just a generalization of definitional equality:

- $<>=<>$, $\Sigma, c{:}K = \Sigma', c{:}K'$ iff $\Sigma = \Sigma'$ and $\vdash_\Sigma K = K'$
- $\vdash_\Sigma \Gamma, x{:}K = \vdash_{\Sigma'} \Gamma, x{:}K'$ iff $\vdash_\Sigma \Gamma = \vdash_{\Sigma'} \Gamma$ and $\Gamma \vdash_\Sigma K = K'$
- $\Gamma \vdash_\Sigma K = \Gamma' \vdash_{\Sigma'} K'$ iff $\vdash_\Sigma \Gamma = \vdash_{\Sigma'} \Gamma'$ and $\Gamma \vdash_\Sigma K = K'$
- $\Gamma \vdash_\Sigma k{:}K = \Gamma' \vdash_{\Sigma'} k'{:}K'$ iff $\Gamma \vdash_\Sigma K\ \ kind = \Gamma' \vdash_{\Sigma'} K'\ \ kind$ and $\Gamma \vdash_\Sigma k = k'{:}K$
- $\Gamma \vdash_\Sigma k = l{:}K = \Gamma' \vdash_{\Sigma'} k' = l'{:}K'$ iff $\Gamma \vdash_\Sigma K\ \ kind = \Gamma' \vdash_{\Sigma'} K'\ \ kind$ and $\Gamma \vdash_\Sigma k = k'{:}K$ and $\Gamma \vdash_\Sigma l = l'{:}K$
- $\Gamma \vdash_\Sigma A \leq_c B = \Gamma' \vdash_{\Sigma'} A' \leq_{c'} B'$ iff $\Gamma \vdash_\Sigma A{:}Type = \Gamma' \vdash_{\Sigma'} A'{:}Type$ and $\Gamma \vdash_\Sigma B{:}Type = \Gamma' \vdash_{\Sigma'} B'{:}Type$ and $\Gamma \vdash_\Sigma c{:}(A)B = \Gamma' \vdash_{\Sigma'} c'{:}(A')B'$

We consider the analogous notation for judgements of the form $\vdash \Gamma_0;<>$, $\vdash \Gamma_0;\Gamma$ and $\Gamma_0;\Gamma \vdash J$. We will say that the judgements are definitionally equal in a certain system if all the corresponding definitional equality judgements are derivable in that system.

According to [21, 33], if we add coercive subtyping and coercive definition rules from Figure 12 in the appendix to a system enriched with a coherent set of subtyping judgements $\mathcal{C}_\Sigma$, any derivation tree in $\Pi[\mathcal{C}_\Sigma]^{;}$ can be translated to a derivation tree in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ (that is a derivation tree that does not use coercive application and definition rules - CA_1, CA_2 and CD) and their conclusions are definitionally equal. We aim to use that result to prove that for any judgement using a coherent signature in $\Pi_{S,\leq}$, there exists a judgement definitionally equal to it in $\Pi^{0K}_{S,\leq}$. For this we shall first prove that $\mathcal{C}_\Sigma$ is coherent in the sense of the definition 17 if Σ is coherent in the sense of the definition 1. To prove this we need to describe the possible contexts at the lefthand side of ; in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ used to infer coercive subtyping judgements.

We first prove a theorem used throughout the section which allows us to argue about judgements in $\Pi^{0K}_{S,\leq}$ and judgements in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ interchangeably. We start by presenting a lemma representing the base case and then the theorem appears as an extension easily proven by induction. The lemma is not required to prove the theorem but it gives a better intuition. The theorem essentially states that for contexts at the lefthand side of ; obtained by interleaving membership entries in a the image through $\Gamma.$ of a valid signature Σ or its prefixes give judgements in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ corresponding to judgements in $\Pi^{0K}_{S,\leq}$. We will see later that all the contexts at the lefthand side of ; in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ are in fact obtained by interleaving membership entries in prefixes of Σ.

▶ **Lemma 21.** *Let $\Sigma \equiv \Sigma_1, \Sigma_2, \Sigma_3$ be a valid signature in $\Pi^{0K}_{S,\leq}$ then, for any c, K and Σ'_1, Σ'_2 s.t. $\Sigma_1 = \Sigma'_1$ and $\Sigma_1, \Sigma_2 = \Sigma'_1, \Sigma'_2$ the following hold:*

- $\vdash \Gamma_{\Sigma'_1}, c{:}K, \Gamma_{\Sigma'_2}; <>$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ *iff* $\Sigma'_1, c{:}K, \Sigma'_2$ *valid is derivable in* $\Pi^{0K}_{S,\leq}$
- $\vdash \Gamma_{\Sigma'_1}, c{:}K, \Gamma_{\Sigma'_2}; \Gamma$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ *iff* $\vdash_{\Sigma'_1, c:K, \Sigma'_2} \Gamma$ *is derivable in* $\Pi^{0K}_{S,\leq}$
- $\Gamma_{\Sigma'_1}, c{:}K, \Gamma_{\Sigma'_2}; \Gamma \vdash J$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ *iff* $\Gamma \vdash_{\Sigma'_1, c:K, \Sigma'_2} J$ *is derivable in* $\Pi^{0K}_{S,\leq}$.

Proof. By induction on the structure of derivation. ◀

Mainly by repeatedly applying the previous lemma (except for the case when we weaken with the empty sequence, which is straight forward by induction on the structure of derivations) we can prove:

▶ **Theorem 22** (Equivalence for $\Pi^{0K}_{S,\leq}$). *Let $\Sigma \equiv \Sigma_1, ..., \Sigma_n$ bea valid signature in $\Pi^{0K}_{S,\leq}$ then, for any $1 \leq k \leq n$, for any $\{\Gamma_i\}_{i\in\{0..k\}}$ sequences free of subtyping entries and and $\Sigma'_1, ..., \Sigma'_k$ s.t. $\Sigma_1, ..., \Sigma_k = \Sigma'_1, ..., \Sigma'_k$ for any $i \in \{1..k\}$ the following hold:*

- $\vdash \Gamma_0, \Gamma_{\Sigma'_1}, \Gamma_1, \Gamma_{\Sigma'_2}, \Gamma_2, ..., \Gamma_{k-1}\Gamma_{\Sigma'_k}, \Gamma_k; <>$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ *if and only if* $\Gamma_0, \Sigma'_1, \Gamma_1, \Sigma'_2, \Gamma_2, ..., \Gamma_{k-1}, \Sigma'_k, \Gamma_k$ *valid is derivable in* $\Pi^{0K}_{S,\leq}$
- $\vdash \Gamma_0, \Gamma_{\Sigma'_1}, \Gamma_1, \Gamma_{\Sigma'_2}, \Gamma_2, ..., \Gamma_{k-1}\Gamma_{\Sigma'_k}, \Gamma_k; \Gamma$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ *if and only if* $\vdash_{\Gamma_0, \Sigma'_1, \Gamma_1, \Sigma'_2, \Gamma_2, ..., \Gamma_{k-1}\Sigma'_k, \Gamma_k} \Gamma$ *is derivable in* $\Pi^{0K}_{S,\leq}$
- $\Gamma_0, \Gamma_{\Sigma'_1}, \Gamma_1, \Gamma_{\Sigma'_2}, \Gamma_2, ..., \Gamma_{k-1}\Gamma_{\Sigma'_k}, \Gamma_k; \Gamma \vdash J$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ *if and only if* $\Gamma \vdash_{\Gamma_0, \Sigma'_1, \Gamma_1, \Sigma'_2, \Gamma_2, ..., \Gamma_{k-1}\Sigma'_k, \Gamma_k} J$ *is derivable in* $\Pi^{0K}_{S,\leq}$.

Now we aim to prove that we do not introduce any new subtyping entries in $\Pi^{0K}_{S,\leq}$ by weakening (up to definitional equality). Note that, for this, it is essential that the weakening rules do not add subtyping entries. More precisely, in the following Lemma we prove a form of strengthening, which roughly says that by strengthening the assumptions of a subtyping judgement, we can still derive it(up to definitional equality).

▶ **Lemma 23.** *Let $\Sigma \equiv \Sigma_1, \Sigma_2$ a valid signature in $\Pi^{0K}_{S,\leq}$, for any c, K, $\Sigma'_1 = \Sigma_1$ and $\Sigma'_1, \Sigma'_2 = \Sigma_1, \Sigma_2$, if $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} A \leq_c B$ is derivable in $\Pi^{0K}_{S,\leq}$ then there exists A', c', B' such that $\vdash_\Sigma A' \leq_{c'} B'$, $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} A = A'{:}Type$, $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} B = B'{:}Type$ and $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} c = c'{:}(A)B$ derivable in $\Pi^{0K}_{S,\leq}$.*

Proof. By induction on the structure of derivation of $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} A \leq_c B$. If it comes from transitivity with the premises $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} A \leq_{c_1} C$ and $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} C \leq_{c_2} B$ then by IH, there exist $A', C', c'_1, C'', B', c'_2$ s.t. $\vdash_\Sigma A' \leq_{c'_1} C'$ and $\vdash_\Sigma C'' \leq_{c'_2} B'$ and $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} A = A'{:}Type$, $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} B = B'{:}Type$, $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} C = C'{:}Type$, $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} C = C''{:}Type$, $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} c_1 = c'_1{:}(A)C$ and $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} c_2 = c'_2{:}(C)B$. By transitivity of equality we have $\Gamma \vdash_{\Sigma'_1, k:K, \Sigma'_2} C' = C''{:}Type$. By lemma 16 we have that

$\Gamma \vdash_{erase(\Sigma'_1,k:K,\Sigma'_2)} C' = C'':Type$ is derivable in Π_S. Similarly, because $\vdash_\Sigma C':Type$ and $\vdash_\Sigma C'':Type$ we have that $\vdash_{erase(\Sigma'_1,k:K,\Sigma'_2)} C':Type$ and $\vdash_{erase(\Sigma'_1,k:K,\Sigma'_2)} C'':Type$ are derivable in Π_S. From Strengthening Lemma([11]) which holds for Π_S we have that $\vdash_{erase(\Sigma)} C' = C'':Type$. Again, by 16 we obtain $\vdash_\Sigma C' = C'':Type$. At last, we can apply congruence and transitivity $\vdash_\Sigma A' \leq_{c'_2 \circ c'_1} B'$.

Let us now consider the dependent product rule

$$\frac{\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} A'' \leq_{c_1} A' \quad \Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} B', B'' : (A')Type \quad \Gamma, x{:}A' \vdash_{\Sigma'_1,k:K,\Sigma'_2} B'(x) \leq_{c_2[x]} B''(x)}{\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} \Pi(A', B') \leq_c \Pi(A'', B'' \circ c_1)}$$

with $A \equiv \Pi(A', B')$, $B \equiv \Pi(A'', B'' \circ c_1)$ and $c \equiv [F : \Pi(A', B')]\lambda(A'', B'' \circ c_1, [x{:}A'']c_2[x](app(A', B', F, c_1(x))))$.

By IH, there exist $A''_0, A'_0, c'_1, B'_0, B''_0, c'_2$ s.t. $\vdash_\Sigma A''_0 \leq_{c'_1} A'_0$, $\vdash_\Sigma B' \leq_{c'_2} B''$ and $\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} A'' = A''_0:Type$, $\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} A' = A'_0:Type$, $\Gamma, x{:}A' \vdash_{\Sigma'_1,k:K,\Sigma'_2} B''(x) = B''_0:Type$, $\Gamma, x{:}A' \vdash_{\Sigma'_1,k:K,\Sigma'_2} B'(x) = B'_0:Type$, $\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} c_1 = c'_1:(A'')A'$ and $\Gamma, x{:}A' \vdash_{\Sigma'_1,k:K,\Sigma'_2} c_2(x) = c'_2:(B'(x))B''(x):Type$.

We apply dependent product rule for the case when types are constants and obtain $\vdash A'_0 \longrightarrow B'_0 \leq'_c A''_0 \longrightarrow B''_0$ with $c' \equiv [F : A'_0 \longrightarrow B'_0][x{:}A''_0](c''_2(F(c'_1(x))))$.

By normal equality rules for dependent product and its terms we have that $\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} A'_0 \longrightarrow B'_0 = \Pi(A', B')$, $\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} A''_0 \longrightarrow B''_0 = \Pi(A'', B'')$ and $\Gamma \vdash_{\Sigma'_1,k:K,\Sigma'_2} c = c':(\Pi(A', B'))\Pi(A'', B'')$ ◀

By repeatedly applying the previous lemma we obtain

▶ **Corollary 24.** *For Σ valid derivable in $\Pi^{0K}_{S,\leq}$, $\Sigma \equiv \Sigma_1, ..., \Sigma_n$, for any $\{\Gamma_i\}_{i\in\{0..n\}}$ sequences free of subtyping entries and $\{\Sigma'_i\}_{i\in\{1..n\}}$ s.t. $\Sigma_1, ..., \Sigma_i = \Sigma'_1, ..., \Sigma'_i$ for any $i \in \{1..n\}$, if $\Gamma \vdash_{\Gamma_0,\Sigma_1,\Gamma_1,\Sigma_2,\Gamma_2,...,\Gamma_{n-1}\Sigma_n,\Gamma_n} A \leq_c B$ is derivable in $\Pi^{0K}_{S,\leq}$ then there exists A', c', B' s.t. $\vdash_\Sigma A' \leq_{c'} B'$, $\Gamma \vdash_{\Gamma_0,\Sigma_1,\Gamma_1,\Sigma_2,\Gamma_2,...,\Gamma_{n-1}\Sigma_n,\Gamma_n} A = A':Type$, $\Gamma \vdash_{\Gamma_0,\Sigma_1,\Gamma_1,\Sigma_2,\Gamma_2,...,\Gamma_{n-1}\Sigma_n,\Gamma_n} B = B':Type$ and $\Gamma \vdash_{\Gamma_0,\Sigma_1,\Gamma_1,\Sigma_2,\Gamma_2,...,\Gamma_{n-1}\Sigma_n,\Gamma_n} c = c':(A)B$ derivable in $\Pi^{0K}_{S,\leq}$.*

Next we prove that weakening does not break coherence:

▶ **Lemma 25.** *For Σ valid in $\Pi^{0K}_{S,\leq}$, if $\Sigma \equiv \Sigma_1, \Sigma_2, \Sigma_3$ is coherent, for any Σ'_1, Σ'_2 s.t. $\Sigma_1 = \Sigma'_1$ and $\Sigma_1, \Sigma_2 = \Sigma'_1, \Sigma'_2$, for any c, K s.t. $\Sigma'_1, c{:}K, , \Sigma'_2$ is valid, $\Sigma'_1, c{:}K, , \Sigma'_2$ coherent.*

Proof. Let us consider the derivable judgements $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2} A \leq_c B$ and $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2} A \leq_d B$. Then we know from Lemma 24 that there exist A', B', A'', B'', c', d' s.t. $\vdash_{\Sigma_1,\Sigma_2} A' \leq_{c'} B'$, $\vdash_{\Sigma_1,\Sigma_2} A'' \leq_{d'} B''$, $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2} A' = A:Type$, $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2} B'' = B:Type$, $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2 s} B'' = B:Type$ $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2} c = c':(A)B$ and $\Gamma \vdash_{\Sigma'_1,c:K,,\Sigma'_2} d = d':(A)B$. As in the proof of the previous lemma, using Lemma 16 and Strengthening Lemma from [11] we have that $\vdash_{\Sigma_1,\Sigma_2} A' = A'':Type$, $\vdash_{\Sigma_1,\Sigma_2} B' = B'':Type$. By congruence we have that $\vdash_{\Sigma_1,\Sigma_2} A' \leq_{d'} B'$ is derivable in $\Pi^{0K}_{S,\leq}$. If Σ is coherent then any prefix of it $\Sigma_1, ..., \Sigma_k$ is coherent so $\vdash_{\Sigma_1,\Sigma_2} c' = d':(A')B'$. Further, by weakening and Lemma 15, we have the desired result. ◀

By repeatedly applying the previous lemma we obtain:

▶ **Lemma 26.** *For Σ valid in $\Pi^{0K}_{S,\leq}$, if $\Sigma \equiv \Sigma_1, ..., \Sigma_n$ is coherent, for any $1 \leq k \leq n$, for any $\{\Gamma_i\}_{i\in\{0..k\}}$ sequences free of subtyping entries, for any $\{\Sigma'_i\}_{i\in\{0..k\}}$ s.t. $\Sigma_1, ..., \Sigma_i = \Sigma'_1, ..., \Sigma'_i$ for any $i \in \{1..k\}$ s.t. $\Gamma_0, \Sigma_1, \Gamma_1, \Sigma_2, \Gamma_2, ..., \Gamma_{k-1}, \Sigma_k, \Gamma_k$ is valid, $\Gamma_0, \Sigma_1, \Gamma_1, \Sigma_2, \Gamma_2, ..., \Gamma_{k-1}, \Sigma_k, \Gamma_k$ is coherent.*

Finally, the following lemma describes the relation between parts of the context at the lefthand side of the ; of judgements in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ and Σ. This is a very important result for proving the coherence of $\mathcal{C}_\Sigma$ based on the coherence of Σ. It states that any such context is in fact obtained from weakening of a prefix of Σ. In addition from this Lemma, because all the derivable judgements in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ that are not in $\Pi^{;}$ are subtyping judgements, we have as a consequence that all the judgements of $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ are equivalent to judgements in $\Pi^{0K}_{S,\leq}$.

▶ **Lemma 27.** *For Σ a valid signature in $\Pi^{0K}_{S,\leq}$, for any derivable judgement $\Gamma';\Gamma \vdash J$ in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ there exists a partition of $\Sigma \equiv \Sigma_1, ..., \Sigma_n$, $1 \leq k \leq n$, $\Gamma_0, ..., \Gamma_k$ free of subtyping entries and $\Sigma'_1, ..., \Sigma'_k$ with $\Sigma'_1, ..., \Sigma'_i = \Sigma'_1, ..., \Sigma'_i$ for any $1 \leq i \leq k$ s.t. $\Gamma' \equiv \Gamma_{\Gamma_0,\Sigma_1,\Gamma_1,...,\Sigma_k,\Gamma_k}$*

Proof. By induction on the structure of derivation of the judgement in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$. We only prove a case for third point when the judgement is $\Gamma';\Gamma \vdash A \leq_c B$. The only nontrivial case is when the judgements follows from weakening. Let us assume it comes from a derivation tree ending with

$$\frac{\Gamma'_1, \Gamma'_2; \Gamma \vdash A \leq_c B \quad \Gamma'_1; <>\vdash K \ \ kind}{\Gamma'_1, c{:}K, \Gamma'_2; \Gamma \vdash A \leq_c B}$$

with $\Gamma' \equiv \Gamma_1, c{:}K, \Gamma_2$. By IH we know that there exists a partition of $\Sigma \equiv \Sigma_1, ..., \Sigma_n$ and $1 \leq k \leq n$ and $\Gamma_0, ..., \Gamma_k$ and $\Sigma'_1, ..., \Sigma'_k$ with $\Sigma'_1, ..., \Sigma'_i = \Sigma'_1, ..., \Sigma'_i$ for any $1 \leq i \leq k$ s.t. $\Gamma'_1, \Gamma'_2 \equiv \Gamma_{\Gamma_0,\Sigma'_1,\Gamma_1,...,\Sigma'_k,\Gamma_k}$ with $\Gamma \vdash_{\Gamma_0,\Sigma'_1,\Gamma_1...,\Sigma'_k,\Gamma_k} A \leq_c B$. Let us consider the case when $\Gamma'_1 \equiv \Gamma_{\Gamma_0,\Sigma'_1,\Gamma_1,...,\Gamma_{i-1},\Sigma^{1\prime}_i}$ and $\Gamma'_2 \equiv \Gamma_{\Sigma^{2\prime}_i,\Gamma_i,...,\Sigma'_k,\Gamma_k}$. With $\Sigma'_i \equiv \Sigma^{1\prime}_i, \Sigma^{2\prime}_i$ for some $1 \leq i \leq k$. We consider the partition of $\Sigma \equiv \Sigma_1, ..., \Sigma^1_i, \Sigma^2_i, ..., \Sigma_n$ s.t. $\Sigma'_1, ..., \Sigma^{1\prime}_i, \Sigma^{2\prime}_i, ..., \Sigma'_n$ $\Sigma_1, ..., \Sigma_l = \Sigma'_1, ..., \Sigma'_l$ for any $l \in 1..i-1$, $\Sigma_1, ..., \Sigma^1_i = \Sigma'_1, ..., \Sigma^{1\prime}_i$, $\Sigma_1, ..., \Sigma^1_i, \Sigma^2_i = \Sigma'_1, ..., \Sigma^{1\prime}_i, \Sigma^{2\prime}_i$ and $\Sigma_1, ..., \Sigma_l = \Sigma'_1, ..., \Sigma'_l$ for any $l \in i+1..n$ and $\Gamma_0, ..., \Gamma_{i-1}, c{:}K, \Gamma_i, ..., \Gamma_k$ s.t. $\Gamma' = \Gamma_{\Gamma_0,\Sigma_1,...,\Gamma_{i-1},\Sigma^1_i,c:K,\Sigma^2_i,\Gamma_i,...,\Sigma_k,\Gamma_k}$. ◀

The next lemma refers to the ability to argue about coherence of a set of coercive subtyping judgements corresponding to a signature.

▶ **Theorem 28** (Equivalence of Coherence). *Let Σ be a valid signature in $\Pi^{0K}_{S,\leq}$. Then Σ is coherent in the sense of the Definition 1 iff $\mathcal{C}_\Sigma$ is coherent for $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ in the sense of the Definition 17.*

Proof. Only if: Let $\Gamma';\Gamma \vdash A \leq_c B$ and $\Gamma';\Gamma \vdash A \leq_d B$ be derivable in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$. From Lemma 27, it follows that there exists a partition of $\Sigma \equiv \Sigma_1, ..., \Sigma_n$ and $1 \leq k \leq n$ and $\Gamma_0, ..., \Gamma_k$ s.t. $\Gamma' = \Gamma_{\Gamma_0,\Sigma_1,...,\Sigma_k,\Gamma_k}$. If Σ is coherent, then $\Gamma_0, \Sigma_1, ..., \Sigma_k, \Gamma_k$ is coherent (from Lemma 26). From Theorem 22, $\Gamma';\Gamma \vdash A \leq_c B$ and $\Gamma';\Gamma \vdash A \leq_d B$ are derivable in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ iff $\Gamma \vdash_{\Gamma_0,\Sigma_1,...,\Sigma_k,\Gamma_k} A \leq_c B$ and $\Gamma \vdash \Gamma_0, \Sigma_1, ..., \Sigma_k, \Gamma_k A \leq_d B$ are derivable in $\Pi^{0K}_{S,\leq}$. From coherence here we have $\Gamma \vdash_{\Gamma_0,\Sigma_1,...,\Sigma_k,\Gamma_k} c = d{:}(A)B$ which is derivable in $\Pi^{0K}_{S,\leq}$ iff $\Gamma';\Gamma \vdash c = d{:}(A)B$ is derivable in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ (again by Theorem Theorem 22).

If: By Theorem 22, $\Gamma \vdash_\Sigma A \leq_c B{:}Type$ and $\Gamma \vdash_\Sigma A \leq_d B{:}Type$ are derivable in $\Pi^{0K}_{S,\leq}$ iff $\Gamma_\Sigma;\Gamma \vdash A \leq_c B$ and $\Gamma_\Sigma;\Gamma \vdash A \leq_d B$ are derivable in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$. Because $\mathcal{C}_\Sigma$ is coherent, $\Gamma_\Sigma;\Gamma \vdash c = d{:}(A)B$ is derivable in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ which happens iff $\Gamma \vdash_\Sigma c = d{:}(A)B$ is derivable in $\Pi^{0K}_{S,\leq}$ ◀

To prove that the system $\Pi_{S,\leq}$ is well behaved we first prove that it is well behaved when all the signatures considered are valid in the restricted system $\Pi^{0K}_{S,\leq}$. First we prove another equivalence lemma for this situation.

▶ **Theorem 29** (Equivalence for $\Pi_{S,\leq}$). *For Σ valid in $\Pi^{0K}_{S,\leq}$, the following hold:*

- $\vdash \Gamma_\Sigma;\Gamma$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}$ *iff* $\vdash_\Sigma \Gamma$ *is derivable in* $\Pi_{S,\leq}$

- $\Gamma_\Sigma; \Gamma \vdash J$ *is derivable in* $\Pi[\mathcal{C}_\Sigma]^{;}$ *iff* $\Gamma \vdash_\Sigma J$ *is derivable in* $\Pi_{S,\leq}$.

Proof. By induction on the structure of derivation. ◀

The following theorem shows that the system we defined here is well behaved and that every coercive subtyping application is really just an abbreviation.

▶ **Lemma 30.** *If a valid signature* Σ *in* $\Pi^{0K}_{S,\leq}$ *is coherent the following hold:*

1. *If* $\vdash_\Sigma \Gamma$ *is derivable in* $\Pi_{S,\leq}$ *then there exists* Γ' *s.t.* $\vdash_\Sigma \Gamma'$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *and* $\vdash_\Sigma \Gamma = \Gamma'$ *is derivable in* $\Pi_{S,\leq}$.
2. *If* $\Gamma \vdash_\Sigma J$ *is derivable in* $\Pi_{S,\leq}$ *then there exists* Γ', J' *s.t.* $\Gamma' \vdash_\Sigma J'$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *and* $\vdash_\Sigma \Gamma = \Gamma'$ *and* $\Gamma \vdash_\Sigma J = J'$ *are derivable in* $\Pi_{S,\leq}$.

Proof. By Theorem 28, since Σ is coherent in, $\mathcal{C}_\Sigma$ is coherent. If we look at the last case, by Theorem 29, $\Gamma \vdash_\Sigma J$ is derivable in $\Pi_{S,\leq}$ iff $\Gamma_\Sigma; \Gamma \vdash J$ is derivable in $\Pi[\mathcal{C}_\Sigma]^{;}$. From [21, 33] we know that, when $\mathcal{C}_\Sigma$ is coherent, any derivation tree of $\Gamma_\Sigma; \Gamma \vdash J$ can be translated into a derivation tree in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ which concludes with the judgement definitionally equal to $\Gamma_\Sigma; \Gamma \vdash J$. So let us consider one such derivation tree, its translation and the definitionally equal conclusion $\Gamma_\Sigma; \Delta \vdash J'$ ($\vdash \Gamma_\Sigma; <>$ is already derivable in $\Pi[\mathcal{C}_\Sigma]^{;}_{0K}$ so by inspecting the definition of the translation in [21, 33] we observe that Γ_Σ will not be changed by the translation). We have $\vdash \Gamma_\Sigma; \Gamma = \Gamma_\Sigma; \Delta$ and $\Gamma_\Sigma; \Gamma \vdash J = J'$ are derivable in $\Pi[\mathcal{C}_\Sigma]^{;}$. From Lemma 29 we know that in this case $\vdash_\Sigma \Gamma = \Delta$ and $\Gamma \vdash_\Sigma J = J'$ are derivable in $\Pi_{S,\leq}$ so the desired derivable judgement is simply $\Delta \vdash_\Sigma J'$. ◀

Note that the previous theorem covers the well-behavedness of judgements derived under a signature that is valid in $\Pi^{0K}_{S,\leq}$. We now prove further that any signature valid in $\Pi_{S,\leq}$ is definitionally equal to a signature valid in $\Pi^{0K}_{S,\leq}$, then because of signature replacement we have that any judgement derivable in in $\Pi_{S,\leq}$ is definitionally equal to a judgement derivable in $\Pi^{0K}_{S,\leq}$.

▶ **Lemma 31.** *For any signature* Σ *valid in* $\Pi_{S,\leq}$ *there exists* Σ' *valid in* $\Pi^{0K}_{S,\leq}$ *s.t.* $\Sigma = \Sigma'$ *in* $\Pi_{S,\leq}$

Proof. By induction on the length of Σ. We assume $\Sigma = \Sigma_0, c{:}K$. By IH we have that there exists Σ'_0 valid in $\Pi^{0K}_{S,\leq}$ s.t. $\Sigma_0 = \Sigma'_0$. By repeatedly applying signature replacement to $\vdash_{\Sigma_0} K\ kind$ we have $\vdash_{\Sigma'_0} K\ kind$ is derivable in $\Pi_{S,\leq}$. By Theorem 30, we have that there exists K' s.t. $\vdash_{\Sigma'_0} K'\ kind$ is derivable in $\Pi^{0K}_{S,\leq}$ with $\vdash_{\Sigma'_0} K = K'$. That means we can derive, in $\Pi^{0K}_{S,\leq}$, $\Sigma'_0, c{:}K'\ valid$. Going back with context replacement we also have $\vdash_{\Sigma_0} K = K'$ derivable, so $\Sigma'_0, c{:}K'$ is the signature we are looking for. ◀

We finish this section with the following theorem:

▶ **Theorem 32.** *If a valid signature* Σ *in* $\Pi_{S,\leq}$ *is coherent the following hold:*

1. *If* $\vdash_\Sigma \Gamma$ *is derivable in* $\Pi_{S,\leq}$ *then there exists* Σ', Γ' *s.t.* $\vdash_{\Sigma'} \Gamma'$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *and* $\Sigma = \Sigma'$ *and* $\vdash_\Sigma \Gamma = \Gamma'$ *are derivable in* $\Pi_{S,\leq}$.
2. *If* $\Gamma \vdash_\Sigma J$ *is derivable in* $\Pi_{S,\leq}$ *then there exists* Σ', Γ', J' *s.t.* $\Gamma' \vdash_{\Sigma'} J'$ *is derivable in* $\Pi^{0K}_{S,\leq}$ *and* $\Sigma = \Sigma'$, $\vdash_\Sigma \Gamma = \Gamma'$ *and* $\Gamma \vdash_\Sigma J = J'$ *are derivable in* $\Pi_{S,\leq}$.

Proof. According to the Lemma 31 there exist Σ' valid in $\Pi^{0K}_{S,\leq}$ s.t. $\Sigma = \Sigma'$. If we consider the last point, by signature replacement $\Gamma \vdash_{\Sigma'} J$ is derivable $\Pi_{S,\leq}$. Because Σ' valid in $\Pi^{0K}_{S,\leq}$, we can apply the Lemma 30 to obtain $\Gamma' \vdash_{\Sigma'} J'$ s.t. $\vdash_{\Sigma'} \Gamma = \Gamma'$ and $\Gamma \vdash_{\Sigma'} J = J'$ are derivable in $\Pi_{S,\leq}$. Again by signature replacement $\vdash_\Sigma \Gamma = \Gamma'$ and $\Gamma \vdash_\Sigma J = J'$. ◀

General Subtyping Rules

$$\frac{\Gamma \Vdash K = K'}{\Gamma \Vdash K \le K'} \qquad \frac{\Gamma \Vdash K \le K' \quad \Gamma \Vdash K' \le K''}{\Gamma \Vdash K \le K''} \qquad \frac{\Gamma \Vdash A = B{:}Type}{\Gamma \Vdash A \le B{:}Type}$$

$$\frac{\Gamma \Vdash A \le B{:}Type \quad \Gamma \Vdash B \le C{:}Type}{\Gamma \Vdash A \le C{:}Type}$$

Subtyping in Contexts

$$\frac{\Gamma \Vdash A{:}Type \quad \alpha \notin FV(\Gamma)}{\Gamma, \alpha \le A \ \ valid} \qquad \frac{\Gamma, \alpha \le A, \Gamma' \ \ valid}{\Gamma, \alpha \le A, \Gamma' \Vdash \alpha{:}Type} \qquad \frac{\Gamma, \alpha \le A, \Gamma' \ \ valid}{\Gamma, \alpha \le A, \Gamma' \Vdash \alpha \le A{:}Type}$$

Type Lifting and Subtyping

$$\frac{\Gamma \Vdash A \le B{:}Type}{\Gamma \Vdash El(A) \le El(B)} \qquad \frac{\Gamma \Vdash k{:}K \quad \Gamma \Vdash K \le K'}{\Gamma \Vdash k{:}K'} \qquad \frac{\Gamma \Vdash k = k'{:}K \quad \Gamma \Vdash K \le K'}{\Gamma \Vdash k = k'{:}K'}$$

Dependent Product

$$\frac{\begin{array}{c}\Gamma \Vdash \Pi(A,B){:}Type \quad \Gamma \Vdash \Pi(A',B'){:}Type \\ \Gamma \Vdash A' \le A{:}Type \quad \Gamma, x{:}A' \Vdash B \le B'{:}Type\end{array}}{\Gamma \Vdash \Pi(A,B) \le \Pi(A',B'){:}Type}$$

Figure 7 Inference Rules for $\Pi_{\le}$

Further, according to the lemma 16, the derivability of any nonsubtyping judgement in $\Pi^{0K}_{S,\le}$ is equivalent to the derivability of a judgement in Π_S and any subtyping judgement in $\Pi^{0K}_{S,\le}$ implies a judgement in Π_S.

3 Embedding Subsumptive Subtyping

In this section, we consider how to embed subsumptive subtyping into coercive subtyping. To this end, we consider a subtyping system which is a reformulation of the one studied by [2] and show how it can be faithfully embedded into our system of coercive subtyping.

We consider a system analogous to Π_S with the difference that we leave out the signatures. The types of judgements in this system are Γ *valid*, $\Gamma \Vdash K$ *kind*, $\Gamma \Vdash k{:}K$, $\Gamma \Vdash K = K'$ and $\Gamma \Vdash k = k'{:}K$ syntactically analogous to $\vdash_{<>} \Gamma$, $\Gamma \vdash_{<>} K$ *kind*, $\Gamma \vdash_{<>} k{:}K$, $\Gamma \vdash_{<>} K = K'$ respectively $\Gamma \vdash_{<>} k = k'{:}K$, baring rules analogous to the ones in the appendix and Figure 2. Note that there will be no Signature Validity and Assumption rules as there are no signatures. On top of these judgements we add $\Gamma \Vdash A \le B$ *type* and $\Gamma \Vdash K \le K'$ obtained with the rules from Figure 7. Besides the ordinary variables in Π, we allow Γ to have subtyping variables like $\alpha \le A$. We name this extension $\Pi_{\le}$.

$\Pi_{\le}$ is the subsumptive subtyping system specified in LF that corresponds to the system $\lambda P_{\le}$ in [2]. There are some subtle differences between Edinburgh LF (λP) [12] and the logical framework LF we use (eg, the η-rule holds for the latter but not the former), but they are irrelevant to the point we are trying to show: the subsumptive subtyping system can be faithfully embedded in the coercive subtyping system.

Once we introduced this system we will proceed by giving an interpretation of it in the coercive subtyping system that we introduced in section 2, namely we will show that this calculus can be faithfully embedded in the coercive subtyping one.

We mentioned that, in this system, an important thing to note is how placing subtyping entries in contexts interferes with abstraction and hence dependent types, specifically, the abstraction is not allowed at the lefthand side of subtyping entries. We will give a mapping that sends the contexts with subtyping entries in the subsumptive system to signatures in the coercive system, prove that these signatures are coherent, and, finally, that we can embed the subsumptive subtyping system into the coercive subtyping system via this mapping. We are motivated, on the one hand by giving a coercive subtyping system in which we can represent this subsumptive system and at the same time allowing abstraction happen freely and on the other hand by the fact that we could not employ coercive subtyping in context as we could make coherent contexts incoherent with substitution. For example if $\alpha_1 \leq_{c_1} A, \alpha_2 \leq_{c_2} A, \Gamma$ is a coherent context (i.e. under this context any two coercions between the same types are equal), by substitution we can obtain the incoherent context $\alpha \leq_{c_1} A, \alpha \leq_{c_2} A, [\alpha_1/\alpha][\alpha_2/\alpha]\Gamma$.

We will assume that Δ is an arbitrary context in $\Pi_{\leq}$. We can also assume without loss of generality that $\Delta \equiv \Delta_1, \alpha_1 \leq A_1, ..., \Delta_n, \alpha_n \leq A_n, \Delta_{n+1}$, where $\{\alpha_i \leq A_i\}_{i=\overline{1,n}}$ are all of the subtyping entries of Δ. If Δ_{n+1} is free of subtyping entries we can abstract over its entries freely but the abstraction is obstructed by $\alpha_n \leq A_n$ for the entire prefix. We move this prefix, together with the obstructing entry to the signature using constant coercions $\Sigma_\Delta = \Delta_1, \alpha_1{:}Type, c_1{:}(\alpha_1)A_1, \alpha_1 \leq_{c_1} A_1{:}Type, ..., \Delta_n, \alpha_n{:}Type, c_n{:}(\alpha_n)A_n, \alpha_n \leq_{c_n} A_n{:}Type$. We map the left Δ_{n+1} to a context. This way we translate $\Delta \equiv \Delta_1, \alpha_1 \leq A_1, ..., \Delta_n, \alpha_n \leq A_n, \Delta_{n+1} \vdash J$ in $\Pi_{\leq}$ to $\Delta_{n+1} \vdash_{\Sigma_\Delta} J$ in $\Pi_{S,\leq}$, with Σ_Δ as above. In the rest of the section we shall prove that mapping subsumptive subtyping entries in context to constant coercions in signature is indeed adequate. For this, we first prove that such a signature is coherent.

▶ **Lemma 33.** *For any valid context Δ in $\Pi_{\leq}$, Σ_Δ is coherent w.r.t. $\Pi_{S,\leq}$.*

Proof. We need to show that, in $\Pi_{S,\leq}$, if we have $\Gamma \vdash_{\Sigma_\Delta} T_1 \leq_c T_2$ and $\Gamma \vdash_{\Sigma_\Delta} T_1 \leq_{c'} T_2$, then $c = c'{:}(T_1)T_2$. There are two cases:

1. $T_1 \equiv \alpha$ is a constant. By the validity of Δ, we have that, if $\alpha_i \leq A_i$ and $\alpha_j \leq A_i$ are two different subtyping entries in Δ, then $\alpha_i \neq \alpha_j$, therefore, if $\alpha_i \leq_{c_i} A_i$ and $\alpha_j \leq_{c_j} A_i$ are two different coercions in Σ_Δ, then necessarily, $\alpha_i \neq \alpha_j$.
2. $T_1 \equiv \Pi(A, B)$ and $T_2 \equiv \Pi(A'', B'')$. In this case the non trivial situation is:

$$\frac{\Gamma \vdash_\Sigma \Pi(A,B) \leq_{c_1} C \quad \Gamma \vdash_\Sigma C \leq_{c_2} \Pi(A'',B'')}{\Gamma \vdash_\Sigma \Pi(A,B) \leq_{c_2 \circ c_1} \Pi(A'',B'')}$$

and C is equal to dependent product too. What we need to show is that applying dependent product rule followed by transitivity leads to the same coercion as applying transitivity first and then the dependent product rule. Namely that, for some A', B' s.t.

$$\frac{\Gamma \vdash_{\Sigma_\Delta} A'' \leq_{c_2} A' \leq_{c_1} A \quad \Gamma \vdash_{\Sigma_\Delta} B \leq_{d_1} B' \leq_{d_2} B''}{\Gamma \vdash_{\Sigma_\Delta} \Pi(A,B) \leq_{e_1} \Pi(A',B') \leq_{e_2} \Pi(A'',B'')}$$

where, for $F{:}A \longrightarrow B$ and $G{:}\Pi(A',B')$, $e_1(F) = \lambda[x'{:}A']d_1(app(F, c_1(x')))$ and $e_2(G) = \lambda[x''{:}A'']d_2(app(G, c_2(x'')))$ applying transitivity rule, first to A, A', A'' and to B, B', B'' and then to $\Pi(A,B)$, $\Pi(A',B')$, $\Pi(A'',B'')$ results in the same coercion, that is:

$$\begin{aligned} e_2 \circ e_1 &= e_2(e_1(F)) \\ &= \lambda[x''{:}A'']d_2(app(e_1(F), c_2(x''))) \\ &=_\beta \lambda[x''{:}A'']d_2(d_1(app(F, c_1(c_2(x''))))) \\ &= d_2 \circ d_1(app(F, c_1(c_2(x'')))) \end{aligned}$$

◀

Notation If $\Gamma \vdash_\Sigma k{:}K$ and $\Gamma \vdash_\Sigma K \leq_c K'$ are derivable in $\Pi_{S,\leq}$, we write $\Gamma \vdash_\Sigma k :: K'$.

In what follows we essentially prove that we can represent the previously introduced subsumptive subtyping system in our system with coercive subtyping in signatures, meaning that we can argue about the former system with the sematic richness of the latter.

▶ **Theorem 34** (Embedding Subsumptive Subtyping). *Let Δ and Γ be valid contexts in $\Pi_\leq$, such that Γ does not contain any subtyping entries. Then we have:*
1. *If Δ, Γ is valid in $\Pi_\leq$ then $\vdash_{\Sigma_\Delta} \Gamma$ valid in $\Pi_{S,\leq}$.*
2. *If $\Delta, \Gamma \Vdash K$ kind, then $\Gamma \vdash_{\Sigma_\Delta} K$ kind in $\Pi_{S,\leq}$.*
3. *If $\Delta, \Gamma \Vdash K = K'$, then $\Gamma \vdash_{\Sigma_\Delta} K = K'$ in $\Pi_{S,\leq}$.*
4. *If $\Delta, \Gamma \Vdash k{:}K$, then $\Gamma \vdash_{\Sigma_\Delta} k{::}K$ in $\Pi_{S,\leq}$.*
5. *If $\Delta, \Gamma \Vdash k = k'{:}K$, then $\Gamma \vdash_{\Sigma_\Delta} k = k'{::}K$ in $\Pi_{S,\leq}$.*
6. *If $\Delta, \Gamma \Vdash A \leq B{:}Type$ then $\Gamma \vdash_{\Sigma_\Delta} A \leq_c B{:}Type$ for some coercion $c{:}(A)B$ in $\Pi_{S,\leq}$.*
7. *If $\Delta, \Gamma \Vdash K \leq K'$, then $\Gamma \vdash_{\Sigma_\Delta} K \leq_c K'$ for some $c{:}(K)K'$ in $\Pi_{S,\leq}$.*

Proof. The proof proceeds by induction on derivations for all the points of the theorem and we only exhibit it for the sixth point here and in particular when the last rule in the derivation tree is the one for the dependent product. We have by IH that, for $\Gamma \vdash_{\Sigma_\Delta} \Pi(A, B){::}Type$ and $\Gamma \vdash_{\Sigma_\Delta} \Pi(A', B'){::}Type$ we have $\Gamma \vdash_{\Sigma_\Delta} A' \leq_c A{:}Type$ and $\Gamma, x{:}A' \vdash_{\Sigma_\Delta} B \leq_{c'} B'{:}Type$. Note that, if $K \leq_c Type$, then $K \equiv Type$, so $\Gamma \vdash_{\Sigma_\Delta} \Pi(A, B){::}Type$ is equivalent to $\Gamma \vdash_{\Sigma_\Delta} \Pi(A, B){:}Type$, and $\Gamma \vdash_{\Sigma_\Delta} \Pi(A', B'){::}Type$ with $\Gamma \vdash_{\Sigma_\Delta} \Pi(A', B'){:}Type$, hence we can directly apply the rule for dependent product in $\Pi_{S,\leq}$ to obtain $\Gamma \vdash_{\Sigma_\Delta} \Pi(A, B) \leq_d \Pi(A', B'){:}Type$ where, for $F{:}\Pi(A, B)$, $d(F) = \lambda[x{:}A']c'(app(F, c(x)))$. ◀

4 Intuitive Notions of Subtyping as Coercion

In this section, we consider two case studies of how intuitive notions of subtyping may be considered in the framework of coercive subtyping. The first is about type universes in type theory and the second is about how injectivity of coercions may play a crucial role in modelling intuitive notions of subtyping.

4.1 Subtyping between Type Universes

A universe is a type of types. One may consider a sequence of universes indexed by natural numbers $U_0 : U_1 : U_2 : ...$ and $U_0 \leq U_1 \leq U_2 \leq ...$

Martin Löf [23] introduced two styles of universes in type theory: the Tarski-style and the Russell-style. The Tarski-style universes are semantically more fundamental but the Russell-style universes are easier to use in practice. In fact, the Russell-style universes are a special case of subsumptive subtyping, which is incompatible with the idea of canonical objects. As observed by the second author in [18], the two styles of universes are not equivalent and the Russell-style universes can be emulated by Tarski-style universes with coercive subtyping and this allows one to reason about Russell universes with the semantic richness of Tarski universes, but without the overhead of their syntax.

Problem with Russell-style Universes. We extend the subsumptive subtyping system $\Pi_\leq$ with Russell-style universes by adding the following rules ($i \in \omega$):

$$\frac{\Gamma \ valid}{\Gamma \Vdash U_i : Type} \qquad \frac{\Gamma \Vdash A : U_i}{\Gamma \Vdash A : Type} \qquad \frac{\Gamma \ valid}{\Gamma \Vdash U_i : U_{i+1}} \qquad \frac{\Gamma \ valid}{\Gamma \Vdash U_i \leq U_{i+1}}$$

and the rules for the Π-types:

$$\frac{\Gamma \Vdash A : U_i \quad \Gamma \Vdash B : (A)U_i}{\Gamma \Vdash \Pi(A, B) : U_i}$$

Unfortunately, as mentioned in the introduction, this straightforward formulation of universes does not satisfy the properties of canonicity or subject reduction if one adopts the standard notation of terms with full type information. For instance, the term $\lambda X{:}U_1.Nat$, where $Nat : U_0$, would be represented as $\lambda(U_1, [\_{:}U_1]U_0, [\_{:}U_1]Nat)$, but this term, which is of type $U_0 \to U_0$ (by subsumption, since $U_1 \to U_0 \le U_0 \to U_0$ by contravariance), is not definitionally equal to any canonical term which is of the form $\lambda(U_0, ...)$. As explained in the introduction, if one used terms with less type information (eg, pairs (a, b), as in HoTT [32], rather than $pair(A, B, a, b)$, there would be incompatible types of the same term and that would cause problems in type-checking.

Tarski-style Universes with Coercive Subtyping. The Tarski-style universes are introduced into $\Pi_{S,\le}$ by adding the following rules ($i \in \omega$):

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma U_i : Type} \qquad \frac{\Gamma \vdash_\Sigma a : U_i}{\Gamma \vdash_\Sigma T_i(a) : Type} \qquad \frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma t_{i+1} : (U_i)U_{i+1}}$$

where t_{i+1} are the lifting operators,

$$\frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma u_i : U_{i+1}} \qquad \frac{\vdash_\Sigma \Gamma}{\Gamma \vdash_\Sigma T_{i+1}(u_i) = U_i : Type}$$

where u_i is the name of U_i in U_{i+1}, together with the following rule for the names of Π-types:

$$\frac{\Gamma \vdash_\Sigma a : U_i \quad \Gamma,\ x : T_i(a) \vdash_\Sigma b(x) : U_i}{\Gamma \vdash_\Sigma \pi_i(a, b) : U_i}$$

The following equations also need to be satisfied:
$T_{i+1}(t_{i+1}(a)) = T_i(a){:}Type$
$\Gamma \vdash_\Sigma T_i(\pi_i(a, b)) = \Pi(T_i(a), [x{:}T_i(a)]T_i(b(x))) : Type$
$\Gamma \vdash_\Sigma t_{i+1}(\pi_i(a, b)) = \pi_{i+1}(t_{i+1}(a), [x{:}T_i(a)]t_{i+1}(b(x))) : U_{i+1}$
Furthermore, crucially, the lifting operators t_{i+1} are now declared as coercions by asking that all the signatures start with the prefix $\Sigma_i \equiv U_0 \le_{t_0} U_1,\ \ ...,\ U_{i-1} \le_{t_i} U_i$ where i is bigger than the largest universe index that is used in an application.

Use of Coercion-based Tarski-style Universes. If universes are specified in the Tarski-style as above with the lifting operators declared as coercions, together with several notational conventions (eg, T_i is omitted, u_i is identified with U_i, etc.), they can now be used easily in Russell-style. The lifting operators are not seen (implicit) by the users. In particular, in this setting, all the Russell-style universe rules become derivable. Theorem 34 can now be extended in such a way that the Russell-style universes are faithfully emulated by the Tarski-style universes with coercive subtyping.

4.2 Injectivity and Constructor Subtyping

In subsumptive subtyping, $A \le B$ means that A is directly embedded in B. Intuitively, this may imply that, for a and a' in A, if the images of them are not equal in B, then they are not equal in A, either. If we consider coercive subtyping $A \leq_c B$, this would imply that c is injective in the sense that $c(a) = c(a')$ implies that $a = a'$. In this section, we shall formally discuss this issue in the context of representing intuitive subtyping notions by means of coercions.

We shall consider constructor subtyping, studied by [4], in which an (inductive) type is considered to be a subtype of another if the latter has more constructors than the former. More precisely we shall discuss the example they start from, namely Even Numbers($Even$) being a subtype of Natural Numbers (Nat) with the argument that the constructors of $Even$ are 0 and successor of Odd, where Odd is given by the constructor successor of $Even$. Then, in Nat the successor constructor is overloaded to a lifting of these constructors as well. Formally they write:

```
datatype Odd = S of Even and Even = 0
    |S of Odd
datatype Nat  = 0
    |S of Nat
    |S of Odd
    |S of Even
```

The phenomenon we want to discuss here is injectivity, in particular the one related to Leibnitz equality. Leibnitz equality is defined as follows: $x = y$ if for any predicate P, $P(x) \iff P(y)$. We denote by $x =_A y$ for some type A the Leibnitz equality between x and y related to a certain domain. Then, we have injectivity of subtyping if, given $x =_{Nat} y$, with $x, y{:}Even$ it is the case that $x =_{Even} y$. Namely, whether for any predicate $Q{:}Even \longrightarrow Prop$, it is the case that $Q(x) \iff Q(y)$. For this it is enough to show that any predicate $Q{:}Even \longrightarrow Prop$ admits a lifting $Q'{:}Nat \longrightarrow Prop$ s.t. for any $x{:}Even, Q'(x) \Longrightarrow Q(x)$. We can easily define such a Q' as follows:

```
Q'(x) =  Q(0) if x = 0
         Q(S(n)) if x = S of n:Odd
         true if x = S of n:Even
         true if x = S of n:Nat
```

Injectivity of the embedding holds here but it is not granted in coercive subtyping. For functions $f{:}(x{:}A)B$ we denote $injective(f) = \forall x, y{:}A.f(x) =_B f(y) \longrightarrow x =_A y$. A function f is then injective if $\exists p{:}injective(f)$.

▶ **Definition 35.** We say a coercion $\vdash_{\Sigma_0, A \leq_c B, \Sigma_1} A \leq_c B$ is **injective with respect to** $=_B$ if there exist p s.t. $\vdash_\Sigma p{:}injective(c)$ is derivable.

For a constant coercions (namely of the form $\vdash_{\Sigma_0, c:(A)B, \Sigma_1, A \leq_c B, \Sigma_2, \Sigma_3} A \leq_c B$) we can add the assumption that they are injective $\vdash_{\Sigma_0, c:(A)B, \Sigma_1, A \leq_c B, \Sigma_2, p:injective(c), \Sigma_3} A \leq_c B$. If we embed a subsumptive subtyping that propagates an equality from a type throughout its subtypes, we represent it as a constant coercion, thus, all we need to do is add the assumption that a coercion is injective. It is obvious that the transitivity and congruence preserve the injectivity property.

An example of noninjective coercions is if we think of Nat and $Even$ as follows

```
Inductive Nat : Type :=
    | O : Nat
    | S : Nat -> Nat.
Inductive even : Nat -> Prop :=
    | O1 : even O
```

```
    | O2 : even O
    | S1 : forall n1, even n1 -> even (S (S n1)).
Inductive Even := pair{n:Nat; e:even n}. Definition proj1(ev:Even) :=
    match
        ev with pair n e => n
        end.
Coercion proj1 : Even >-> Nat.
```

Note that the definition of *Even* changed and we refer to it as a feature of the natural numbers rather than as a subset. In order for a natural number to be even we require a proof of that.

The reason this coercion is not injective is that we can have two different proofs that 4 is even $p_1, p_2{:}even4$, and hence, two different pairs $(4, p_1), (4, p_2){:}Even$, both of them being mapped to the same $4{:}Nat$. Enforcing injectivity here is similar to enforcing proof irrelevance.

5 Conclusion and Future Work

In this paper, we have developed a new calculus of coercive subtyping and shown that subsumptive subtyping can be faithfully embedded or represented in the calculus. The idea of representing coercive subtyping relations in signatures has achieved a balance between obtaining a powerful (and practical) calculus to capture intuitive notions of subtyping and keeping the resulting calculus simple enough for meta-theoretic studies.

We intend to extend the calculus to a richer type theory like Martin-Löf's type theory or UTT where you have rich inductive types. We do not see any difficulty in doing so, but of course, studies are needed to confirm this.

Specifying subtyping relations in signatures has changed the nature of 'basic subtyping relations' as studied in the earlier setting of coercive subtyping. The earlier setting allows parameterised coercions such as $n{:}Nat \vdash Vect(Nat, n) \leq_{c(n)} List(Nat)$, which instantiates, in particular, to $\vdash Vect(Nat, 3) \leq_{c(3)} List(Nat)$. Note that here we don't use *parameterised* in the sense of Coq Proof Assistant. This new system does not cover this kind of coercions at this point. It would be interesting to study a new mechanism to introduce parameterised coercions by means of entries in signatures.

Acknowledgements. Thanks go to Sergei Soloviev for extremely helpful remarks on this work during his visit to Royal Holloway and the anonymous referees for their helpful comments.

References

1 The Agda proof assistant (version 2). http://appserv.cs.chalmers.se/users/ulfn/wiki/agda.php, 2008.

2 D. Aspinall and A. Compagnoni. Subtyping dependent types. *Theoretical Computer Science*, 266:273–309, 2001.

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

4 G. Barthe and M. J. Frade. Constructor subtyping. *Lecture Notes in Computer Science*, 1576:109–127, 1999.

5 Gustavo Betarte and Alvaro Tasistro. Extension of Martin-Löf's type theory with record types and subtyping. *Twenty-five Years of Constructive Type Theory*, 1998.

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

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

8 S. Chatzikyriakidis and Z. Luo. Natural language inference in Coq. *J. of Logic, Language and Information.*, 23(4):441–480, 2014.

9 S. Chatzikyriakidis and Z. Luo. *Formal Semantics in Modern Type Theories.* ISTE/Wiley, 2018. (to appear).

10 The Coq Development Team. *The Coq Proof Assistant Reference Manual (Version 8.3), INRIA*, 2010.

11 Healfdene Goguen. *A Typed Operational Semantics for Type Theory.* PhD thesis, University of Edinburgh, 1994.

12 R. Harper, F. Honsell, and G. Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40:143–184, 1993.

13 Y. Luo. *Coherence and Transitivity in Coercive Subtyping.* PhD thesis, University of Durham, 2005.

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

15 Z. Luo. Coercive subtyping in type theory. In *Proc. of CSL'96, the 1996 Annual Conference of the European Association for Computer Science Logic, Utrecht. LNCS 1258*, page draft., 1996.

16 Z. Luo. Coercive subtyping. *Journal of Logic and Computation*, 9, 1999.

17 Z. Luo. Formal semantics in modern type theories with coercive subtyping. *Linguistics and Philosophy*, 35(6):491–513, 2012.

18 Z. Luo. Notes on Universes in Type Theory (for a talk given at Institute of Advanced Studies), 2012. URL: `https://uf-ias-2012.wikispaces.com/file/view/LuoUniverse.pdf`.

19 Z. Luo. Formal semantics in modern type theories: Is it model-theoretic, proof-theoretic, or both? (invited talk). In Nicholas Asher and Sergei Soloviev, editors, *Logical Aspects of Computational Linguistics*, volume 8535 of *Lecture Notes in Computer Science*, pages 177–188. Springer Berlin Heidelberg, November 2014.

20 Z. Luo and R. Pollack. LEGO Proof Development System: User's Manual. LFCS Report ECS-LFCS-92-211, Dept of Computer Science, Univ of Edinburgh, 1992.

21 Z. Luo, S. Soloviev, and T. Xue. Coercive subtyping: theory and implementation. information and computation. *Information and Computation*, 223:18–42, 2013.

22 Zhaohui Luo and Fjodor Part. Subtyping in type theory: Coercion contexts and local coercions, TYPES 2013, Toulouse. 2013.

23 P. Martin-Löf. *Intuitionistic Type Theory.* Bibliopolis, 1984.

24 The Matita proof assistant. Available from: `http://matita.cs.unibo.it/`, 2008.

25 J. C. Mitchell. Coercion and type inference. In *POPL'83*, 1983.

26 Bengt Nordström, Kent Petersson, and Jan M. Smith. *Programming in Martin-Löf's Type Theory: An Introduction.* Clarendon Press, New York, NY, USA, 1990.

27 Benjamin C. Pierce. Bounded quantification is undecidable. In *Information and Computation*, pages 305–315, 1993.

28 J. Reynolds. The meaning of types: From intrinsic to extrinsic semantics. *BRICS Report Series RS-00-32*, 2000.

29 John C. Reynolds. Using category theory to design implicit conversions and generic operators. *Semantics-Directed Compiler Generation 1980*, Lecture Notes in Computer Science 94, 1980.

30 A. Saïbi. Typing algorithm in type theory with inheritance. *POPL'97*, 1997.

31 S. Soloviev and Z. Luo. Coercion completion and conservativity in coercive subtyping. *Annals of Pure and Applied Logic*, 113(1-3):297–322, 2002.

32 Univalent Foundations Program. *Homotopy Type Theory: Univalent Foundations of Mathematics*. Institute for Advanced Study, 2013.

33 Tao Xue. *Theory and Implementation of Coercive Subtyping*. PhD thesis, Royal Holloway University of London, 2013.

A Rules of $\Pi[C]^{;}$

The rules of $\Pi[C]^{;}$ consists of those in Figures 8, Figure 9, Figure 10, Figure 11 and Figure 12.

Validity of Signature/Contexts, Assumptions

$$\frac{}{\vdash \langle\rangle} \qquad \frac{\Sigma; <>\vdash K\ kind \quad c \notin dom(\Sigma)}{\vdash \Sigma, c{:}K} \qquad \frac{\vdash \Sigma, c{:}K, \Sigma'; \Gamma}{\Sigma, c{:}K, \Sigma'; \Gamma \vdash c{:}K}$$

$$\frac{\vdash \Sigma}{\vdash \Sigma; \langle\rangle} \qquad \frac{\Sigma; \Gamma \vdash K\ kind \quad x \notin dom(\Sigma) \cup dom(\Gamma)}{\vdash \Sigma; \Gamma, x{:}K} \qquad \frac{\vdash \Sigma; \Gamma, x{:}K, \Gamma'}{\Sigma; \Gamma, x{:}K, \Gamma' \vdash x{:}K}$$

Weakening

$$\frac{\Sigma, \Sigma'; \Gamma \vdash J \quad \Sigma; <>\vdash K\ kind \quad c \notin dom(\Sigma, \Sigma')}{\Sigma,\ c{:}K,\ \Sigma'; \Gamma \vdash J}$$

$$\frac{\Sigma; \Gamma, \Gamma' \vdash J \quad \Sigma; \Gamma \vdash K\ kind \quad x \notin dom(\Gamma, \Gamma')}{\Sigma; \Gamma, x{:}K, \Gamma' \vdash J}$$

Equality Rules

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

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

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

Context Replacement

$$\frac{\Sigma_0, c{:}L, \Sigma_1; \Gamma \vdash J \quad \Sigma_0 \vdash L = L'}{\Sigma_0, c{:}L', \Sigma_1; \Gamma \vdash J} \qquad \frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash J \quad \Sigma; \Gamma_0 \vdash K = K'}{\Sigma; \Gamma_0, x{:}K', \Gamma_1 \vdash J}$$

Substitution Rules

$$\frac{\vdash \Sigma; \Gamma_0, x{:}K, \Gamma_1 \quad \Sigma; \Gamma_0 \vdash k{:}K}{\vdash \Sigma; \Gamma_0, [k/x]\Gamma_1}$$

$$\frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash K'\ \ kind \quad \Sigma; \Gamma_0 \vdash k{:}K}{\Sigma; \Gamma_0, [k/x]\Gamma_1 \vdash [k/x]K'\ \ kind} \qquad \frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash L = L' \quad \Sigma; \Gamma_0 \vdash k{:}K}{\Sigma; \Gamma_0, [k/x]\Gamma_1 \vdash [k/x]L = [k/x]L'}$$

$$\frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash k'{:}K' \quad \Sigma; \Gamma_0 \vdash k{:}K}{\Sigma; \Gamma_0, [k/x]\Gamma_1 \vdash [k/x]k'{:}[k/x]K'} \qquad \frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash l = l'{:}K' \quad \Sigma; \Gamma_0 \vdash k{:}K}{\Sigma; \Gamma_0, [k/x]\Gamma_1 \vdash [k/x]l = [k/x]l'{:}[k/x]K'}$$

$$\frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash K'\ \ kind \quad \Sigma; \Gamma_0 \vdash k = k'{:}K}{\Sigma; \Gamma_0, [k/x]\Gamma_1 \vdash [k/x]K' = [k'/x]K'} \qquad \frac{\Sigma; \Gamma_0, x{:}K, \Gamma_1 \vdash l{:}K' \quad \Sigma; \Gamma_0 \vdash k = k'{:}K}{\Sigma; \Gamma_0, [k/x]\Gamma_1 \vdash [k/x]l = [k'/x]l{:}[k/x]K'}$$

Dependent Product Kinds

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

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

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

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

The kind Type

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

Figure 8 Inference Rules for $LF^{;}$

$$\frac{\Sigma;\Gamma \vdash A : Type \quad \Sigma;\Gamma, x{:}A \vdash B(x) : Type}{\Sigma;\Gamma \vdash \Pi(A,B) : Type}$$

$$\frac{\Sigma;\Gamma \vdash A : Type \quad \Sigma;\Gamma \vdash B : (A)Type \quad \Sigma;\Gamma \vdash f : (x{:}A)B(x)}{\Sigma;\Gamma \vdash \lambda(A,B,f) : \Pi(A,B)}$$

$$\frac{\Sigma;\Gamma \vdash g : \Pi(A,B) \quad \Sigma;\Gamma \vdash a : A}{\Sigma;\Gamma \vdash app(A,B,g,a) : B(a)}$$

$$\frac{\begin{array}{c}\Sigma;\Gamma \vdash A : Type \quad \Sigma;\Gamma \vdash B : (A)Type \\ \Sigma;\Gamma \vdash f : (x{:}A)B(x) \quad \Sigma;\Gamma \vdash a : A\end{array}}{\Sigma;\Gamma \vdash app(A,B,\lambda(A,B,f),a) = f(a) : B(a)}$$

Figure 9 Inference Rules for $\Pi^{;}$

Subtyping Rules

$$\frac{\Sigma;\Gamma \vdash A \leq_c B \in \mathcal{C}}{\Sigma;\Gamma \vdash A \leq_c B}$$

Congruence

$$\frac{\Sigma;\Gamma \vdash A \leq_c B : Type \quad \Sigma;\Gamma \vdash A = A' : Type \quad \Sigma;\Gamma \vdash B = B' : Type \quad \Sigma;\Gamma \vdash c = c' : (A)B}{\Sigma;\Gamma \vdash A' \leq_{c'} B' : Type}$$

Transitivity

$$\frac{\Sigma;\Gamma \vdash A \leq_c A' : Type \quad \Sigma;\Gamma \vdash A' \leq_{c'} A'' : Type}{\Sigma;\Gamma \vdash A \leq_{c' \circ c} A'' : Type}$$

Weakening

$$\frac{\Sigma,\Sigma';\Gamma \vdash A \leq_d B : Type \quad \Sigma \vdash K\ kind}{\Sigma,\ c{:}K,\ \Sigma';\Gamma \vdash A \leq_d B : Type} \quad (c \notin dom(\Sigma,\Sigma'))$$

$$\frac{\Sigma;\Gamma,\Gamma' \vdash A \leq_d B : Type \quad \Sigma;\Gamma \vdash K\ kind}{\Sigma;\Gamma, x{:}K, \Gamma' \vdash A \leq_d B : Type} \quad (x \notin dom(\Gamma,\Gamma'))$$

Context Replacement

$$\frac{\Sigma_0, c{:}L, \Sigma_1;\Gamma \vdash A \leq_c B \quad \Sigma_0 \vdash L = L'}{\Sigma_0, c{:}L', \Sigma_1;\Gamma \vdash A \leq_c B} \quad \frac{\Sigma;\Gamma_0, x{:}K, \Gamma_1 \vdash A \leq_c B \quad \Sigma;\Gamma_0 \vdash K = K'}{\Sigma;\Gamma_0, x{:}K', \Gamma_1 \vdash A \leq_c B}$$

Substitution

$$\frac{\Sigma;\Gamma_0, x{:}K, \Gamma_1 \vdash A \leq_c B \quad \Sigma;\Gamma_0 \vdash k{:}K}{\Sigma;\Gamma_0, [k/x]\Gamma_1 \vdash [k/x]A \leq_{[k/x]c} [k/x]B}$$

Identity Coercion

$$\frac{\Sigma;\Gamma \vdash A{:}Type}{\Sigma;\Gamma \vdash A \leq_{[x:A]x} A{:}Type}$$

Dependent Product

$$\frac{\Sigma;\Gamma \vdash A' \leq_{c_1} A : Type \quad \Sigma;\Gamma \vdash B, B' : (A)Type \quad \Sigma;\Gamma, x{:}A \vdash B(x) \leq_{c_2[x]} B'(x) : Type}{\Sigma;\Gamma \vdash \Pi(A,B) \leq_{[F:\Pi(A,B)]\lambda(A', B' \circ c_1, [x:A']c_2[x](app(A,B,F,c_1(x))))} \Pi(A', B' \circ c_1) : Type}$$

Figure 10 Inference Rules for $\Pi[\mathcal{C}]^{;}_{0K}$ (1)

Basic Subkinding Rule and Identity

$$\frac{\Sigma;\Gamma \vdash A \leq_c B{:}Type}{\Sigma;\Gamma \vdash El(A) \leq_c El(B)} \qquad \frac{\Sigma;\Gamma \vdash K\ \ kind}{\Sigma;\Gamma \vdash K \leq_{[x:K]x} K}$$

Structural Subkinding Rules

$$\frac{\Sigma;\Gamma \vdash K_1 \leq_c K_2 \quad \Sigma;\Gamma \vdash K_1 = K_1' \quad \Sigma;\Gamma \vdash K_2 = K_2' \quad \Sigma;\Gamma \vdash c = c'{:}(K_1)K_2}{\Sigma;\Gamma \vdash K_1' \leq_{c'} K_2'}$$

$$\frac{\Sigma;\Gamma \vdash K \leq_c K' \quad \Sigma;\Gamma \vdash K' \leq_{c'} K''}{\Sigma;\Gamma \vdash K \leq_{c' \circ c} K''}$$

$$\frac{\Sigma,\Sigma';\Gamma \vdash K \leq_d K' \quad \Sigma;<>\vdash K_0\ kind}{\Sigma, c{:}K_0, \Sigma';\Gamma \vdash K \leq_d K'} \quad (c \notin dom(\Sigma,\Sigma'))$$

$$\frac{\Sigma;\Gamma,\Gamma' \vdash K \leq_d K' \quad \Sigma;\Gamma \vdash K_0\ kind}{\Sigma;\Gamma, x{:}K_0, \Gamma' \vdash K \leq_d K'} \quad (x \notin dom(\Gamma,\Gamma'))$$

$$\frac{\Sigma_0, c{:}L, \Sigma_1;\Gamma \vdash K \leq_d K' \quad \Sigma_0;<>\vdash L = L'}{\Sigma_0, c{:}L', \Sigma_1;\Gamma \vdash K \leq_d K'} \quad \frac{\Sigma;\Gamma_0, x{:}K, \Gamma_1 \vdash L \leq_d L' \quad \Sigma;\Gamma_0 \vdash K = K'}{\Sigma;\Gamma_0, x{:}K', \Gamma_1 \vdash L \leq_d L'}$$

$$\frac{\Sigma;\Gamma_0, x{:}K, \Gamma_1 \vdash K_1 \leq_c K_2 \quad \Sigma;\Gamma_0 \vdash k{:}K}{\Sigma;\Gamma_0, [k/x]\Gamma_1 \vdash [k/x]K_1 \leq_{[k/x]c} [k/x]K_2}$$

Subkinding for Dependent Product Kind

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

Figure 11 Inference Rules for $\Pi[\mathcal{C}]_{0K}^{;}$ (2)

Coercive Application

$$(CA_1) \qquad \frac{\Sigma;\Gamma \vdash f{:}(x{:}K)K' \quad \Sigma;\Gamma \vdash k_0{:}K_0 \quad \Sigma;\Gamma \vdash K_0 \leq_c K}{\Sigma;\Gamma \vdash f(k_0){:}[c(k_0)/x]K'}$$

$$(CA_2) \qquad \frac{\Sigma;\Gamma \vdash f = f'{:}(x{:}K)K' \quad \Sigma;\Gamma \vdash k_0 = k_0'{:}K_0 \quad \Sigma;\Gamma \vdash K_0 \leq_c K}{\Sigma;\Gamma \vdash f(k_0) = f'(k_0'){:}[c(k_0)/x]K'}$$

Coercive Definition

$$(CD) \qquad \frac{\Sigma;\Gamma \vdash f{:}(x{:}K)K' \quad \Sigma;\Gamma \vdash k_0{:}K_0 \quad \Sigma;\Gamma \vdash K_0 \leq_c K}{\Sigma;\Gamma \vdash f(k_0) = f(c(k_0)){:}[c(k_0)/x]K'}$$

Figure 12 The coercive application and definition rules in $\Pi[\mathcal{C}]^{;}$