ECOLE NORMALE SUPERIEURE

Dependent Type System with Subtyping

Type level Transitivity Elimination

Gang CHEN

LIENS - 96 - 27

Département de Mathématiques et Informatique

CNRS URA 1327

Dependent Type System with Subtyping

Type level Transitivity Elimination

Gang CHEN

LIENS - 96 - 27

December 1996

Laboratoire d'Informatique de l'Ecole Normale Supérieure
45 rue d'Ulm 75230 PARIS Cedex 05

Tel : (33)(1) 44 32 00 00

Adresse électronique : ... @dmi.ens.fr

Dependent Type System with Subtyping

– Type Level Transitivity Elimination

Gang CHEN

December 20, 1996

This work is supported by Programme PRA M4, Association Franco-Chinoise pour la Recherche Scientifique et Technique, and Bourse du Ministère des Affaires Etrangères du Gouvernement Français.

Résume

Le système $\lambda P_{\leq}$ proposé par Aspinall et Compagnani [AC96] est une extension du système de types dépendents λP[Bar92] avec sous-typage. Dans ce rapport, nous étudions une nouvelle approche pour ajouter sous-typage aux types dépendents. On obtient un système $\lambda\Pi_{\leq}$ qui possède un système de sous-typage indépendent du systéme de typage et dont la propriété d'élimination de la transitivité se situe aux au niveau des types, qui distinge cette approche d'autres systèmes de sous-typage, (e.g. $\lambda P_{\leq}$[AC96], $F^{\omega}_{\leq}$[SP94], $F^{\omega}_{\wedge}$[Com94]), où l'élimination de la transitivité est limités aux types en forme normale. Cette technique est très adaptée pour les extensions et les implementations.

Abstract

Dependent type systems have been the basis of many proof development enviroments. In [AC96], a system $\lambda P_{\leq}$ is proposed as a subtyping extension of the dependent type system λP[Bar92] (also called $\lambda\Pi$[Dow95]). $\lambda P_{\leq}$ has nice meta-theoretic properties including subject reduction and decidability, but transitivity elimination is restricted to the β_2 normalized types. In this report, we propose a system $\lambda\Pi_{\leq}$, which is equivalent to $\lambda P_{\leq}$, but it has type level transitivity eliminatioin property. This feature distinguishes our approach from the existing subtyping systems with reduction relations in types. e.g. $\lambda P_{\leq}$[AC96], $F^{\omega}_{\leq}$[SP94], $F^{\omega}_{\wedge}$[Com94], where transitivity elimination only holds for normalized types. Meta-theoretic properties including subject reduction and decidability are established. The system is shown to be equivalent with $\lambda P_{\leq}$ in typing, kinding and context formation. The type checking algorithm is more clear and efficient than $\lambda P_{\leq}$. The technique is suitable for future extensions and real implementations.

Contents

1 Introduction

Recently, there are growing interests on using subtyping in proof checkers [Pfe93, Coq92, Luo96, AC96, Cou95, Bai96, Bar95, Sai96] with the aim of proof reuse and definition reuse. Dependent types are the basis of these proof checkers. Subtyping extension of dependent type system is therefore highly desirable. The system $\lambda P_{\leq}$, developed by Aspinall and Compagnoni[AC96], combines subtyping and first order dependent types (simply called "dependent types" in the sequel), for which fundamental properties, like subject reduction and decidability, have been established.

We had tried to extend $\lambda P_{\leq}$ by overloaded type[CGL95], but, at a time, we felt difficult to progress. The reason is that the subtyping system of $\lambda P_{\leq}$ does not have the transitivity elimination property, besides, typing and subtyping are closely related.

To overcome this problem, we have developed a system $\lambda\Pi_{\leq}$, which has the transitivity elimination property and the system is equivalent to $\lambda P_{\leq}$ in typing, kinding and context formation. Several distinguished features of this work are as follows:

1. Type-level transitivity elimination property.
2. A simple and general $\beta\Gamma$ strong normalization proof.
3. A more efficient subtyping algorithm than $\lambda P_{\leq}$.

Transitivity elimination is an important property for systems with subtypes [LMS95]. From proof theoretical point of view, transitivity elimination is an indispensible step in the proof of subject reduction and decidability of subtyping. When there are type conversion, the transitivity elimination turns out to be difficult. The existing approach is to construct algorithmic systems based on normalized types, and prove the transitivity elimination in the algorithmic system, as can be seen in the studies of $F^{\omega}_{\leq}$[SP94],$F^{\omega}_{\wedge}$[Com94] and $\lambda P_{\leq}$[AC96]. We may say that these systems have transitivity elimination at the level of normalized types. $\lambda\Pi_{\leq}$ is the first system having type level transitivity elimination among subtyping systems with type conversions.

$\beta\Gamma$ strong normalization is a desirable property to prove the termination of subtyping algorithm. In [AC96], this property has not been proved, instead, a measure based on a new type constructor "plus" has been used. [SP94] has proved a $\beta\Gamma$ strong normalization for $F^{\omega}_{\leq}$, which needs alternating β and Γ reduction. Our proof is simple and general.

These two achievements have not only improved the work of [AC96], but also become important progress towards subtyping systems with type conversions, especially subtyping extension of Calculus of Construction (CC). In CC, there are four different reductions mixed together, traditional methods developed in [SP94], [Com94] and [AC96] become difficult to work. With a modification of the method in this paper, we have succeded in subtyping CC[Che96b].

From practical point of view, our subtyping system is more efficient than $\lambda P_{\leq}$ and suitable for implementation in proof checkers.

The following subsections informally discuss dependent types and subtyping, then we sketch our approach.

1.1 Intuitive Introduction of Dependent Types

Dependent types are types depending on terms. They can be viewed as functionals mapping terms to types, or as families of types indexed by terms. A typical example is the collection of product types indexed by natural numbers:

$$\{A, A \times A, A \times A \times A, \cdots, A^n, \cdots\}$$

which can be represented as:

$$\Lambda n{:}nat.P(n)$$

where $P(n)$ denotes the product A^n. The Λ abstraction shows that it is a mapping from naturals to product types:

$$n \ \mapsto A^n$$

Therefore, a product type of length m could be encoded as

$$(\Lambda n{:}nat.P(n))m$$

which reduces to $P(m) = A^m$ by β-reduction.

Expressions like $\Lambda x{:}A.B$ are called type families, or simply families, and can be viewed as type constructors. Applications of type families to terms give rise to types or to other type families. Two problems need to be taken into account:

1. Kinding of type family
2. Type Reduction through type family application

Kinds are assigned to types, type families and applications of type families much the same way as types are assigned to terms. General types have the kind $\star$, the kind of a type family $\Lambda x{:}A.B$ is of form $\Pi x{:}A.K$, where K is the kind of B(under the hypothesis $x : A$). For the previous example, the kinding is

$$\Lambda x{:}nat.P(n) : \Pi x{:}nat.\star$$

Types in a dependent type system are either atomic types, applications of type families (if they have the kind $\star$), or π-types of form $\pi x{:}A.B$, the latter is the counterpart of arrow type $A \rightarrow B$ in simply typed λ-calculus. Therefore, a lambda abstraction $\lambda x{:}A.M$ may have a type $\pi x{:}A.B$.

π-types are not the only kind of types a lambda abstraction can have. Another possible type for $\lambda x{:}A.M$ might be a type of form $(\Lambda y{:}C.\pi x{:}A'.B')N$, which is equivalent to a π-type $\pi x{:}A'[y := N].B'[y := N]$ by β-reduction. Type reductions are taken at two levels: reductions of terms in types or type families; reductions of type family applications, (they are respectively called β_1 and β_2 reductions in [AC96]). Type reduction complicates the study of meta-theoretical properties as the uniqueness of minimal typing is lost: a term has a class of minimal types which are β equivalent.

There are several equivalent presentations of first order dependent type systems. The system studied in this report and in the system $\lambda P_{\leq}$[AC96] are based on the Edinburgh LF type theory [HHP93]. An introduction to dependent types can be found in [Dow95].

From the formula-as-type analogy, the introduction of dependent types brings significant progress with respect to simply typed lambda calculus. In the latter case, only propositional formula can be represented by types, with dependent types, the quantification is also representable. As a result, many logical systems could be encoded in systems based on dependent types, as done in LF, the Edinburgh Logical Framework [HHP93].

1.2 The system $\lambda\Pi$

The system $\lambda\Pi$ (also called λP) is the pure first order dependent type system. It is the core of Edinburgh Logical Framework. Our presentation of $\lambda\Pi$ is mainly based on [HHP93] and [Dow95]. There are four syntactic categoies:

1.	Terms	denoted by M,N, ...
2.	Types and type families	denoted by A,B,C, ...
3.	Kinds	denoted by K,L, ...
4.	Contexts	denoted by Γ

The abstract syntax of the entities is given by the following grammar:

$$\begin{array}{lcl} M & ::= & x \mid \lambda x{:}A.M \mid MM \\ A & ::= & \alpha \mid \pi x{:}A.A \mid \Lambda x{:}A.A \mid AM \\ K & ::= & \star \mid \Pi x{:}A.K \\ \Gamma & ::= & <> \mid \Gamma, x:A \mid \alpha:K \end{array}$$

Explanation:

1. A term can be a term variable (denoted by x,y,z,...), an abstraction or an application
2. A type can be a type variable (denoted by α), a π type of the form $\pi x{:}A.B$ or an type application AM which should be β-convertible to a π-type; A type family, or family, can be a type variable, Λ abstraction of the form $\Lambda x{:}A.M$ or a type application AM;
3. A kind is either a kind constant $\star$ representing the collection of all types or of the form $\Pi x{:}A.K$ which classifies type families of the form $\Lambda x{:}A.B$ where B lives in the kind K. Thus,the general form of a kind would be $\Pi x_1{:}A_1..x_n{:}A_n.\star$ where $n \geq 0$.
4. a context is an ordered list of typing assignments of the form $x : A$ where A is a type or kinding assignments of the form $\alpha : K$.

The above abstract syntax has actually defined preterms, pretypes, prefamilies, prekinds and pre-contexts. Well-formed terms, types, families, kinds and contexts are determined by the following judgements:

$\Gamma \vdash \star$	Γ is a well-formed context
$\Gamma \vdash K$	K is a kind in context Γ
$\Gamma \vdash A : K$	type A has kind K in context Γ
$\Gamma \vdash M : A$	term M has type A in context Γ

We write $\Gamma \vdash J$ for an arbitrary judgement of one of the forms $\Gamma \vdash K, \Gamma \vdash A : K$ or $\Gamma \vdash M : A$. The rules for deriving the judgements in $\lambda\Pi$ are given below:

Context Formation Rules

F-empty $$\frac{}{<> \vdash \star}$$

F-term $$\frac{\Gamma \vdash A : \star \quad x \notin Dom(\Gamma)}{\Gamma, x : A \vdash \star}$$

F-type $$\frac{\Gamma \vdash K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha : K \vdash \star}$$

F-Π $$\frac{\Gamma, x : A \vdash K}{\Gamma \vdash \Pi x{:}A.K}$$

Kinding Rules

$$\text{K-var} \quad \frac{\Gamma \vdash \star \quad \alpha \in Dom(\Gamma)}{\Gamma \vdash \alpha : Kind_\Gamma(\alpha)}$$

$$\text{K-}\pi \quad \frac{\Gamma, x : A \vdash B : \star}{\Gamma \vdash \pi x{:}A.B : \star}$$

$$\text{K-}\Lambda \quad \frac{\Gamma, x : A \vdash B : K}{\Gamma \vdash \Lambda x{:}A.B : \Pi x{:}A.K}$$

$$\text{K-app} \quad \frac{\Gamma \vdash A : \Pi x{:}B.K \quad \Gamma \vdash M : B}{\Gamma \vdash AM : K[x := M]}$$

$$\text{K-conv} \quad \frac{\Gamma \vdash A : K \quad \Gamma \vdash K' \quad K =_\beta K'}{\Gamma \vdash A : K'}$$

Typing Rules

$$\text{T-var} \quad \frac{\Gamma \vdash \star \quad x \in Dom(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\text{T-}\lambda \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A.B}$$

$$\text{T-app} \quad \frac{\Gamma \vdash M : \pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

$$\text{T-conv} \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash A =_\beta B}{\Gamma \vdash M : B}$$

Note that the β-conversion concerns two kinds of β-reduction:

$$(\lambda x{:}A.M)N \to_{\beta_1} M[x := N]$$
$$(\Lambda x{:}A.B)N \to_{\beta_2} B[x := N]$$

1.3 Meta-theoretic Properties of $\lambda\Pi$

$\lambda\Pi$ has the following properties.

Confluence of β reduction holds for pre-expressions. It can be proved in the usual way.

Proposition 1.1 (Church-Rosser property) *Suppose U, U', U'' are pre-expressions. If $U \to_\beta^* U'$ and $U \to_\beta^* U''$, then there exists a pre-expression V such that $U' \to_\beta^* V$ and $U'' \to_\beta^* V$.*

Proposition 1.2 (Subject reduction)

$$\Gamma \vdash J \ \wedge \ J \to_\beta^* J' \quad \Rightarrow \quad \Gamma \vdash J'$$

Proposition 1.3 (β-strong normalization) *For β reduction, we have the following results:*

1. *If $\Gamma \vdash K$, then K is strongly normalizing;*
2. *If $\Gamma \vdash A : K$, then A is strongly normalizing;*
3. *If $\Gamma \vdash M : A$, then M is strongly normalizing.*

Proposition 1.4 (Decidability) *The judgement $\Gamma \vdash J$ is decidable.*

1.4 Motivation on subtyping dependent types

There are several application areas where researchers have discovered the need to combine subtyping and dependent types. Here is an example taken from [Pfe93] concerning economic encoding of logics.

Pfenning [Pfe93] has noticed that at the absence of subtyping, it is cumbersome to deal with the representation of subsets of logical formulas in LF. Consider, for example, the set (or the type) of well formed formula in propositional calculus defined by the syntax:

$$F \ ::= \ Atom \mid \neg F \mid F \wedge F \mid F \vee F \mid F \Rightarrow F \mid$$

and a subset of these formulas defined as:

$$F_1 \ ::= \ Atom \mid \neg F_1 \mid F_1 \vee F_1$$

As pointed out by Pfenning, without subtyping, the representation of subsets of formulas like F_1 is awkward and will lead to inefficient implementation of proof search. To overcome the problem, Pfenning proposed the refinement type, which could be considered as a restricted form of subtyping. With refinement type and bounded intersection type, the above formulas have the following nice declaration

$$\begin{array}{rl} Atom <: F_1 & -- \ Atom \ is \ a \ subsort \ of \ F_1 \ type \ formula \\ F_1 <: F & -- \ F_1 \ typeformula \ is \ a \ subsort \ of \ F \ type \ formula \end{array}$$

$$\begin{array}{rcl} \neg & : & F \rightarrow F \\ \wedge & : & F \rightarrow F \rightarrow F \\ \vee & : & F \rightarrow F \rightarrow F \\ \Rightarrow & : & F \rightarrow F \rightarrow F \end{array}$$

$$\begin{array}{rcl} \neg & : & F_1 \rightarrow F_1 \\ \vee & : & F_1 \rightarrow F_1 \rightarrow F_1 \end{array}$$

where $A <: B$ denotes that A is a subsort of B. There is an implicit use of intersection types in the declaration. The declaration of $\neg$ and $\wedge$ should be transformed to:

$$\begin{array}{rcl} \neg & : & (F \rightarrow F) \cap (F_1 \rightarrow F_1) \\ \vee & : & (F \rightarrow F \rightarrow F) \cap (F_1 \rightarrow F_1 \rightarrow F_1) \end{array}$$

Pfenning's study is within the proof enviroment Elf which is an implementation of Edinburgh LF. Other groups of dependent type theory based proof systems also found the need of using subtyping. The motivation examples are similar. An early work can be found in [Coq92] in the ALT group. The LEGO group, Coq group and Nuprl group are studying implementations of abstract algebra, all of them have proposed extensions of type theory by some sort of subtyping: ZhaoHui Luo [Luo96] has studied the Coercive subtyping extension to LEGO; in the Nuprl group Jason Hickey [HI92] has combined object-calculus and dependent types and proposed a form of subtyping based on the inheritance mechanism of objects; in the Coq group, Courant [Cou95] had attempted an extension of Calculus of Construction by subtyping: $CC_{\leq}$, Saibi [Sai96] has given an algorithm of inheritance on PTS. The strong interests in this area are mainly due to the fact that proof development system are attacking the problem of scale. As said by ZhaoHui Luo[Luo96]: "The lack of useful subtyping mechanisms in dependent type theories with inductive types and the associated proof development systems is one of the obstacles in their applications to large-scale formal development. "

Whilst the researches mentioned above have handled theories richer than $\lambda\Pi$ due to the need of theorem proving practice, the proposed subtyping mechanisms appear more or less restricted from the pure dependent type theory point of view. For example, in the Pfenning's work, the "subsorting" relation may be reformulated as follows (rules concerninig subsorting intersection types and kindings are omitted):

$$\frac{A \leq B \in \Gamma}{\Gamma \vdash A \leq B} \qquad \frac{}{\Gamma \vdash A \leq A} \qquad \frac{\Gamma \vdash A \leq B}{\Gamma \vdash AM \leq BM}$$

$$\frac{\Gamma \vdash A' \leq A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'} \qquad \frac{\Gamma \vdash A \leq C \quad \Gamma \vdash C \leq B}{\Gamma \vdash A \leq B}$$

The presented rules are desirable and the subsorting is decidable. But there are no subtyping relation between type families except from the identity. Furthermore, if types A and B are convertible, they do not have subsorting relation although from the semantic point of view they denote the same set of objects.

1.5 Subtyping in $\lambda P_{\leq}$

The $\lambda P_{\leq}$ [AC96] can be viewed as an enrichment to the Pfenning's subsorting rules. The subtyping between type families have been added, the reflexivity rule is replaced by the conversion rule. The subtyping system is formulated as follows:

S-CONV $$\frac{\Gamma \vdash A, B : K \quad A =_\beta B}{\Gamma \vdash A \leq B}$$

S-TRANS $$\frac{\Gamma \vdash A \leq B \quad \Gamma \vdash B \leq C}{\Gamma \vdash A \leq C}$$

S-VAR $$\frac{\Gamma \vdash \star \quad \alpha \; \textit{bounded in} \; \Gamma}{\Gamma \vdash \alpha \leq \Gamma(\alpha)}$$

S-π $$\frac{\Gamma \vdash A' \leq A, \quad \Gamma, x : A' \vdash B \leq B' \quad \Gamma \vdash \pi x{:}A.B : \star}{\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'}$$

S-Λ $$\frac{\Gamma, x : A \vdash B \leq B' \quad \Gamma \vdash \Lambda x{:}A.B : K}{\Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A.B'}$$

S-APP $$\frac{\Gamma \vdash A \leq B \quad \Gamma \vdash BM : K}{\Gamma \vdash AM \leq BM}$$

In $\lambda P_{\leq}$, the subtyping declaration is introduced to context in the form $\alpha \leq A : K$, meaning that α is a new type variable, which is a subtype of type A with kind K.

The analysis of $\lambda P_{\leq}$ is challenging, principally because it introduces the conversion rule S-CONV guaranteering that β-convertible types occupy the same equivalent class in the subtype relation, and the rule S-APP for subtyping family applications. These rules are responsible for the failure of elimination of transitivity at type level, that is, the rule of transitivity can not be harmlessly taken out from the subtyping system as can be seen from the following example of

transitivity application:

$$\dfrac{\dfrac{C =_\beta \pi x{:}A.B}{\Gamma \vdash C \leq \pi x{:}A.B}\ \text{S-CONV} \quad \dfrac{\Gamma \vdash A' \leq A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'}\ \text{S-}\pi}{\Gamma \vdash C \leq \pi x{:}A'.B'}\ \text{S-TRANS}$$

Subtyping rules have certain kinding requirements in order to make sure that the types and type families involved in subtyping are well-formed. Without them, we may have undesirable subtyping judgements, e.g. $(\pi x{:}A.B)M \leq (\pi x{:}A'.B')M$, where the expressions at two sides of $\leq$ are not well-formed since π-types are not type families and can not be applied to terms. These expressions are members of pretypes.

The kinding conditions are designed so as to ensure the well-formedness of subtyping judgements

$$\Gamma \vdash A \leq B \ \Rightarrow\ \exists K \ \ s.t.\ \ \Gamma \vdash A, B : K$$

The kinding premises in the subtyping rules complicates the study of the system because of the dependencies between typing, kinding, context formation and subtyping.

1.6 Typing in $\lambda P_{\leq}$

The context formation rules, kinding rules and typing rules of $\lambda P_{\leq}$ are as in $\lambda\Pi$ except the introduction of subtyping assumption in the context formation and the subsumption rule in typing.

Context Formation Rules

F-empty $$\dfrac{}{<> \vdash \star}$$

F-term $$\dfrac{\Gamma \vdash A : \star \quad x \notin Dom(\Gamma)}{\Gamma, x : A \vdash \star}$$

F-type $$\dfrac{\Gamma \vdash K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha : K \vdash \star}$$

F-subtype $$\dfrac{\Gamma \vdash A : K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha \leq A : K \vdash \star}$$

F-Π $$\dfrac{\Gamma, x : A \vdash K}{\Gamma \vdash \Pi x{:}A.K}$$

Kinding Rules

$$\text{K-var} \quad \frac{\Gamma \vdash \star \quad \alpha \in Dom(\Gamma)}{\Gamma \vdash \alpha : Kind_\Gamma(\alpha)}$$

$$\text{K-}\pi \quad \frac{\Gamma, x : A \vdash B : \star}{\Gamma \vdash \pi x{:}A.B : \star}$$

$$\text{K-}\Lambda \quad \frac{\Gamma, x : A \vdash B : K}{\Gamma \vdash \Lambda x{:}A.B : \Pi x{:}A.K}$$

$$\text{K-app} \quad \frac{\Gamma \vdash A : \Pi x{:}B.K \quad \Gamma \vdash M : B}{\Gamma \vdash AM : K[x := M]}$$

$$\text{K-conv} \quad \frac{\Gamma \vdash A : K \quad \Gamma \vdash K' \quad K =_\beta K'}{\Gamma \vdash A : K'}$$

Typing Rules

$$\text{T-var} \quad \frac{\Gamma \vdash \star \quad x \in Dom(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\text{T-}\lambda \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A.B}$$

$$\text{T-app} \quad \frac{\Gamma \vdash M : \pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

$$\text{T-sub} \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash A \leq B}{\Gamma \vdash M : B}$$

□

Note that $\lambda P_{\leq}$ has no subkinding rule like:

$$\text{K-sub} \quad \frac{\Gamma \vdash A : K \quad \Gamma \vdash K' \quad K \leq K'}{\Gamma \vdash A : K'}$$

From practical point of view, this simplification seems harmless: the existing subtyping relation is already rich enough. But it brings a great advantage to the development of meta-theoretical properties in $\lambda P_{\leq}$. Because of the circle between typing and subtyping, the proof for subject reduction in [AC96] is divided into two steps, first for β_2 reduction, then for β_1 reduction, via an algorithmic subtyping system, which has transitivity elimination property. Subject reduction for β_2 can be easily proved simply because there are no subkinding. The following subsection gives a detailed analysis and introduces informally our approach to the problem.

1.7 Technical Analysis on Subtyping Dependent Types

The main difficulty in the study of subtyping dependent types is the proof of subject reduction in the case

$$(\lambda x{:}A.M)N \rightarrow_\beta M[x := N]$$

Suppose $\Gamma \vdash (\lambda x{:}A.M)N : D$, we need to prove $\Gamma \vdash M[x := N] : D$. With subtyping, we need to consider, for example, a derivation ended with

$$\dfrac{\dfrac{\dfrac{\Gamma, x : A \vdash M : B'}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A.B'} \quad \Gamma \vdash \pi x{:}A.B' \leq \pi x{:}A'.B}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A'.B}\ subsumption \qquad \Gamma \vdash N : A'}{\Gamma \vdash (\lambda x{:}A.M)N : B[x := N]}$$

where $D = B[x := N]$. From $\Gamma, x : A \vdash M : B'$ we can apply substitution (see Section 3) to get

$$\Gamma \vdash M[x := N] : B'[x := N]$$

To prove $\Gamma \vdash M[x := N] : B[x := N]$, we hope to show that $\Gamma \vdash B'[x := N] \leq B[x := N]$ and apply subsumption. This subtyping assertion should follow from $\Gamma, x : A' \vdash B' \leq B$ by substitution. It is expected that the latter subtyping judgement can be obtained from

$$\Gamma \vdash \pi x{:}A.B' \leq \pi x{:}A'.B$$

Consider the subtyping system of $\lambda P_{\leq}$. There are altogether three possibilities to derive the above judgement: by transitivity rule S-TRANS, by conversion rule S-CONV or by π-subtyping rule S-π. In the last two cases, the judgement $\Gamma, x : A' \vdash B' \leq B$ should be derivable from the assumptions of the rules. With transitivity, its derivability is not straightforward. For example, with a derivation ended by

$$\dfrac{\Gamma \vdash \pi x{:}A.B' \leq (\Lambda y{:}C.D)M \quad \Gamma \vdash (\Lambda y{:}C.D)M \leq \pi x{:}A'.B}{\Gamma \vdash \pi x{:}A.B' \leq \pi x{:}A'.B}\ \text{S-TRANS}$$

it is not clear how the judgement $\Gamma, x : A' \vdash B' \leq B$ could be derived.

Many subtyping systems, e.g. $F_{\leq}$ [CG92],$Co^{\vdash}$[LMS95], have the transitivity elimination property. If the subtyping derivation in our case can be transformed to a derivation without transitivity, as done in those systems, then the derivability of $\Gamma \vdash \pi x{:}A.B' \leq \pi x{:}A'.B$ will imply that of $\Gamma, x : A' \vdash B' \leq B$. Unfortunately, as stated before, transitivity can not be eliminated in $\lambda P_{\leq}$.

To get around this problem, in $\lambda P_{\leq}$, an algorithmic subtyping system is defined on β_2 normal forms (i.e. types without redexes of form $(\Lambda x{:}A.B)N$). Therefore, β_2-equal terms, e.g. $(\Lambda x{:}A.B)M$ and $B[x := M]$, are identified. Transitivity is admissible in algorithmic subtyping system. By the equivalence of the original subtyping system and the algorithmic one:

$$\Gamma \vdash A \leq B \ \Leftrightarrow\ \Gamma \vdash_{\mathcal{A}} A, B : \star \ \wedge\ \Gamma \vdash_{\mathcal{A}} A^{\beta_2} \leq B^{\beta_2}$$

(where $\vdash_{\mathcal{A}}$ denotes provable judgements in the algorithmic system and A^{β_2} denotes the β_2 normal form of A) the subject reduction can be proved. Other fundamental properties can also be derived in $\lambda P_{\leq}$ including confluence, strong normalization and decidability.

1.8 Algorithmic system of $\lambda P_{\leq}$

The algorithmic system of $\lambda P_{\leq}$ is as follows. Note that β_2 strong normalization appear in the rules. It means that this system depends on the β_2 strong normalization property .

Algorithmic Context Formation Rules:

AF-empty $$\frac{}{<> \vdash_{\mathcal{A}} \star}$$

AF-term $$\frac{\Gamma \vdash_{\mathcal{A}} \star \quad \Gamma \vdash_{\mathcal{A}} A : \star \quad x \notin Dom(\Gamma)}{\Gamma, x : A \vdash_{\mathcal{A}} \star}$$

AF-type $$\frac{\Gamma \vdash_{\mathcal{A}} K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha : K \vdash_{\mathcal{A}} \star}$$

AF-subtype $$\frac{\Gamma \vdash_{\mathcal{A}} K \quad \Gamma \vdash_{\mathcal{A}} A : K' \quad K =_\beta K' \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha \leq A : K \vdash_{\mathcal{A}} \star}$$

AF-Π $$\frac{\Gamma, x : A \vdash_{\mathcal{A}} K}{\Gamma \vdash_{\mathcal{A}} \Pi x{:}A.K}$$

Algorithmic Kinding Rules:

AK-var $$\frac{\alpha \in Dom(\Gamma)}{\Gamma \vdash_{\mathcal{A}} \alpha : Kind_\Gamma(\alpha)}$$

AK-π $$\frac{\Gamma \vdash_{\mathcal{A}} A : \star \quad \Gamma, x : A \vdash_{\mathcal{A}} B : \star}{\Gamma \vdash_{\mathcal{A}} \pi x{:}A.B : \star}$$

AK-Λ $$\frac{\Gamma \vdash_{\mathcal{A}} A : \star \quad \Gamma, x : A \vdash_{\mathcal{A}} B : K}{\Gamma \vdash_{\mathcal{A}} \Lambda x{:}A.B : \Pi x{:}A.K}$$

AK-app $$\frac{\Gamma \vdash_{\mathcal{A}} A : \Pi x{:}B.K \quad \Gamma \vdash_{\mathcal{A}} M : B' \quad \Gamma \vdash B'^{\beta_2} \leq B^{\beta_2}}{\Gamma \vdash_{\mathcal{A}} AM : K[x := M]}$$

Algorithmic Typing Rules:

AT-var $$\frac{x \in Dom(\Gamma)}{\Gamma \vdash_{\mathcal{A}} x : \Gamma(x)}$$

AT-λ $$\frac{\Gamma \vdash_{\mathcal{A}} A : \star \quad \Gamma, x : A \vdash_{\mathcal{A}} M : B}{\Gamma \vdash_{\mathcal{A}} \lambda x{:}A.M : \pi x{:}A.B}$$

AT-app $$\frac{\Gamma \vdash_{\mathcal{A}} M : A \quad FLUB_\Gamma(A) \equiv \pi x : B.C \quad \Gamma \vdash_{\mathcal{A}} N : B' \quad \Gamma \vdash B'^{\beta_2} \leq B}{\Gamma \vdash_{\mathcal{A}} MN : C[x := N]}$$

Algorithmic Subtyping Rules:

$$\text{S-}\pi \qquad \frac{\Gamma \vdash A' \leq A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'}$$

$$\text{S-}\Lambda \qquad \frac{A =_{\beta_1} A' \quad \Gamma, x : A \vdash B \leq B'}{\Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A'.B'}$$

$$\text{S-ApR} \qquad \frac{M_1 =_{\beta_1} M_1' \ \cdots \ M_n =_{\beta_1} M_n'}{\Gamma \vdash \alpha M_1..M_n \leq \alpha M_1'..M_n'}$$

$$\text{S-ApT} \qquad \frac{\Gamma \vdash (\Gamma(\alpha)M_1..M_n)^{\beta_2} \leq A}{\Gamma \vdash \alpha M_1..M_n \leq A}$$

where $FLUB_\Gamma(A)$ repeatedly replaces type variables in A by their upper bounds in the context Γ and then do the β_2-normalizations , see [AC96] for details.

The development of meta-theoretical properties in $\lambda P_\leq$ follows a carefully designed order as follows:

1. structural properties including generation of typing, kinding; substitution
2. confluence
3. β_2 subject reduction
4. strong normalization
5. equivalence of algorithmic system with the original system
6. admissiblilities of reflexivity rule, transitivity rule and subtyping application rule
7. generation of subtyping
8. subject reduction
9. decidability

Note that transitivity elimination is restricted to β_2-normalized types and type families. The algorithmic rules are also based on β_2-normalized expressions. As being pointed out by Aspinall and Compagnoni, the β_2-normalizations in type checking are not always necessary and they suggest that the real implementation may make use of only head β_2-normalization.

Proofs of admissibility of transitivity and the decidability of subtyping in $\lambda P_\leq$ are among the hard part of that work and an induction measure based on the β_2-normalization is used in the proof.

1.9 Motivation for this work

The study of $\lambda\Pi_\leq$ begins with two motivations:

1. could subtyping system be defined on pretypes?
2. is there a subtyping system with transitivity elimination at type level?

Proofs in $\lambda P_\leq$ are quite delicate due to the circularity of subtyping and typing system. The $\lambda P_\leq$ is a pure dependent type system with subtyping. In practical application, one will need to add other type constructions such as inductive types, intersection types, overloaded types etc. If we can base subtyping on pretypes, then subtyping rules can be more independent, allowing a relatively easy study.

Transitivity elimination is an important property in the study of subtyping. Proofs of subject reduction and decidability all depends on this property. The algorithmic system of $\lambda P_\leq$ achieves the transitivity elimination at β_2 normalized type level. The main contribution of this system is

two rules S-ApR and S-ApT. The former recovers the conversion at the β_2 normal form level, the later deals with the use of subtyping declaration.

It would be desirable to lift the transitivity elimination to the type level and not restricted to normalized expressions. The technique would be useful in future development of the system.

Now, consider again an example of transitivity application,

$$\frac{\dfrac{\alpha \leq \Lambda x{:}E.F \in \Gamma}{\Gamma \vdash \alpha M \leq (\Lambda x{:}E.F)M} \quad \dfrac{(\Lambda x{:}E.F)M =_\beta F[x := M]}{\Gamma \vdash (\Lambda x{:}E.F)M \leq F[x := M]}}{\Gamma \vdash \alpha M \leq F[x := M]}\ trans$$

In $\lambda P_{\leq}$, this derivation can not be transformed to a transitivity-free one. But, if we introduce a new rule

$$\text{S-ApSL} \quad \frac{\Gamma \vdash B[x := M_1]M_2..M_n \leq C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq C}$$

Then we can have a derivation:

$$\frac{\dfrac{\Gamma \vdash F[x := M] \leq F[x := M]}{\Gamma \vdash (\Lambda x{:}E.F)M \leq F[x := M]}\ \text{S-ApSL}}{\Gamma \vdash \alpha M \leq F[x := M]}\ \text{S-VAR}$$

where the previous transitivity application has been eliminated.

The introduction of the rule S-ApSL is not enough to eliminate all transitivity elimination. Since we do not want conversion rule, we need a rule to derive, for example, $\Gamma \vdash F[x := M] \leq (\Lambda x{:}E.F)M$. Therefore, we introduce a rule S-ApSR, which is symmetric to S-ApSL, to handle the β_2 conversion at right side:

$$\text{S-ApSR} \quad \frac{\Gamma \vdash C \leq B[x := M_1]M_2..M_n}{\Gamma \vdash C \leq (\Lambda x{:}A.B)M_1..M_n}$$

This pair of rules S-ApSR and S-ApSL is the key contribution of our subtyping system to achieve type level transitivity elimination. The remaining subtyping rules come from modifications of the algorithmic system of $\lambda P_{\leq}$. Recall that the algorithmic system of $\lambda P_{\leq}$ is defined on β_2 normal forms, the aim of our modification is to allow the system works at the type level. The rule S-π is kept unchanged. The β_1 conversions in S-Λ and S-ApR are replaced by general β-conversion. Note that, in the original rule of $\lambda P_{\leq}$, the S-Λ does not have the β conversion, but our version of S-Λ can be derived in $\lambda P_{\leq}$ by using transition(S-TRANS) and conversion (S-CONV). Finally, for the rule S-ApT, we do not need the β_2 conversion. The whole set of subtyping rules are presetned at next section and in the Appendix A.

This system can be turned to an ordered rewriting system to check subtyping, (see Section 8 for details). As subtyping is defined on types rather than on β_2 normalized types, there are no β_2 normalizations in the subtyping rules as in $\lambda P_{\leq}$, instead, there are β_2 reductions which usually take less steps than normalization. Therefore, the algorithm is more efficient.

There are no kindings in the subtyping rules, so subtyping is defined on the pretypes and is thus independent of the typing system. The generation of subtyping can be obtained straightforwardly:

$$\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B' \ \Rightarrow\ \Gamma \vdash A' \leq A \ \wedge\ \Gamma, x : A' \vdash B \leq B'$$

while its proof had been the main difficulty in the proof of subject reduction in $\lambda P_{\leq}$.

The typing, kinding and context formation rules are same as in $\lambda P_{\leq}$ except that kinding premises are added to subsumption rule. Algorithmic rules for typing, kinding and context formation are same as those in $\lambda P_{\leq}$ except that there are no β_2 normalization in application rules for typing and kinding.

The development of the meta-theoretical properties follows the sequence:

1. structural properties and Church-Rosser property
2. subject reduction and strong normalization for β_2 head reduction
3. admissibility of reflexivity, transitivity and subtyping application rule
4. generation of typing and subject reduction
5. equivalence between $\lambda\Pi_{\leq}$ and $\lambda P_{\leq}$
6. algorithmic system and decidability of subtyping and typing

One may compare this development with that of $\lambda P_{\leq}$.

In next section, we introduce definitions and basic notations of $\lambda\Pi_{\leq}$ system. As preparation for the proof of admissibility of transitivity, in Section 3, we study some structural properties as well as Church-Rosser property, β_{2h} subject reduction and normalization. β_{2h} strong normalization will be used in the induction measure in the proofs of admissibilities of reflexivity and transitivity, both of which are presented in Section 4. The subject reduction is proved in Section 5. Then, in Section 6, we prove the $\beta\Gamma$ strong normalization, which implies β strong normalization and the termination of subtyping algorithm. The equivalence between $\lambda P_{\leq}$ and $\lambda\Pi_{\leq}$ is eatablished in Section 7. The proof is somewhat convoluted. In Section 8, we present the algorithmic system and prove the decidability of subtyping and typing.

2 Definition of $\lambda\Pi_{\leq}$

As in $\lambda P_{\leq}$, the system $\lambda\Pi_{\leq}$ has four syntactic categories: contexts, kinds, types (including families of types), terms. There are four judgement forms on these notions:

$\Gamma \vdash K$	K is a kind in context Γ
$\Gamma \vdash A : K$	type A has kind K in context Γ
$\Gamma \vdash M : A$	term M has type A in context Γ
$\Gamma \vdash A \leq B$	A is a subtype of B in context Γ

The syntax of *preterms* (denoted by $M, N, ..$), *pretypes* (denoted by $A, B, ..$), *prekinds* (denoted by $K, L, ..$) and *pre-contexts* (denoted by Γ ..) is:

$$\begin{array}{lcl} M & ::= & x \mid \lambda x{:}A.M \mid MM \\ A & ::= & \alpha \mid \pi x{:}A.A \mid \mathbf{\Lambda} x{:}A.A \mid AM \\ K & ::= & \star \mid \Pi x{:}A.K \\ \Gamma & ::= & <> \mid \Gamma, x:A \mid \Gamma, \alpha:K \mid \Gamma, \alpha \leq A:K \end{array}$$

A *pre-expression* (denoted by U or V) is either a preterm, or a pretype, or a prekind. Two distinguished classes of β-reductions are defined over pre-expressions as in $\lambda P_{\leq}$.

$$\begin{array}{l} \mathcal{C}[(\lambda x{:}A.M)N] \rightarrow_{\beta_1} \mathcal{C}[M[x := N]] \\ \mathcal{C}[(\mathbf{\Lambda} x{:}A.B)N] \rightarrow_{\beta_2} \mathcal{C}[B[x := N]] \end{array}$$

($\mathcal{C}[\text{-}]$ indicates a pre-expression with a hole in it).

Define $\rightarrow_{\beta}$ as $\rightarrow_{\beta_1} \cup \rightarrow_{\beta_2}$. We denote by, $\stackrel{*}{\rightarrow}_R$ (*resp.* $=^*_R$) the reflexive and transitive closures of the reduction $\rightarrow_R$ (*resp.* $\rightarrow_R \cup \leftarrow_R$) where $\leftarrow_R$ is the converse relation of $\rightarrow_R$). U^R denotes the R normal form of U.

Context formation, kinding and typing are as in $\lambda P_{\leq}$ except for the subsumption rule in which the kinding checks are added :

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash A \leq B \quad \Gamma \vdash A, B : \star}{\Gamma \vdash M : B} \; subsumption$$

where the notation $\Gamma \vdash A, B : \star$ is the abbreviation for the two judgements $\Gamma \vdash A : \star \quad \Gamma \vdash B : \star$. Similar conventions will be used in the sequel for other judgement forms.

The complete set of context formation rules, kinding rules and typing rules can be found in the Appendix A. The following notations and abbreviations are used:

Notation 2.1 (Notations and abbreviations)

- *$Dom(\Gamma)$ denotes the set of variables declared in context Γ:*
  1. *$x \in Dom(\Gamma)$ if $x : A \in \Gamma$*
  2. *$\alpha \in Dom(\Gamma)$ if $\alpha : K \in \Gamma$ or $\alpha \leq A : K \in \Gamma$*
- *$Kind_\Gamma(\alpha)$ refers to the kind in the declaration of α inside Γ*
  - *$Kind_\Gamma(\alpha) = K$ if $\alpha : K \in \Gamma$ or $\alpha \leq A : K \in \Gamma$*
- *α is said to be* bound *in Γ by type A if $\alpha \leq A : K \in \Gamma$; a variable α can be declared in Γ but* unbound *if $\alpha : K \in \Gamma$*
- $\Gamma(x) \equiv A$ *if* $x : A \in \Gamma$
- $\Gamma(\alpha) \equiv A$ *if* $\alpha \leq A : K \in \Gamma$
- *$Fv(U)$ denotes the set of free variables in U*
- $\Gamma \vdash A, B : K$ *iff* $\Gamma \vdash A : K$ *and* $\Gamma \vdash B : K$
- *$IH \equiv Induction\ Hypothesis$*

No kinding check appears in the subtyping rules so that subtyping is separated from the other judgements. This makes easy to prove many syntactic properties.

In contrast to $\lambda P_\leq$, the subtyping rules are not defined on types but on pretypes.

The subtyping rules are:

$$\text{S-}\pi \qquad \frac{\Gamma \vdash A' \leq A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'}$$

$$\text{S-}\Lambda \qquad \frac{A' =_\beta A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A'.B'}$$

$$\text{S-ApR} \qquad \frac{M_1 =_\beta M_1' \ \cdots \ M_n =_\beta M_n'}{\Gamma \vdash \alpha M_1..M_n \leq \alpha M_1'..M_n'}$$

$$\text{S-ApT} \qquad \frac{\Gamma \vdash \Gamma(\alpha)M_1..M_n \leq A}{\Gamma \vdash \alpha M_1..M_n \leq A}$$

$$\text{S-ApSL} \qquad \frac{\Gamma \vdash B[x := M_1]M_2..M_n \leq C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq C}$$

$$\text{S-ApSR} \qquad \frac{\Gamma \vdash C \leq B[x := M_1]M_2..M_n}{\Gamma \vdash C \leq (\Lambda x{:}A.B)M_1..M_n}$$

The first two rules S-π and S-Λ capture the subtyping relation between function types. The others concerns function applications. Subtyping relation between atomic types can be viewed as special cases of applications. In the previous section, we have discussed the motivation of this system.

The subtyping relation is defined on pretypes, in contrast to subtyping in $\lambda P_\leq$ which is defined on types, as well as to algorithmic subtyping in $\lambda P_\leq$ which is defined on β_2 normal form pretypes.

3 Some meta-theoretical properties

The structural properties in $\lambda\Pi_{\leq}$ are similar to those in $\lambda P_{\leq}$. Sometimes, results are presented in simplified forms.

We write $\Gamma \vdash J$ denote an arbitrary judgement in which J is either K, or $A : K$ or $M : A$. In this report, the notation $Judgement1 \Rightarrow Judgement2$ means that the derivability of the first judgement implies that of the second.

Proofs for most structural properties are usually by induction on the depths of derivations of the concerned judgements: first for the subtyping, then simultaneously for context formation, kinding and typing.

Proofs of many important results require the generation principle which is used to reason about the way that judgements are derived. The generation of typing says that, if the form of a term M is given in a typing judgement $\Gamma \vdash M : A$, then we can tell the form of type A; the generation of kinding allows one to infer from the form of type A, in the judgement $\Gamma \vdash A : K$, the form of K; the generation of subtyping allows one to infer from the form of type A in $\Gamma \vdash A \leq B$ the form of B and vice versa. The proof of the generation for typing is postponed to the Section 5 since it uses the admissibility of transitivity, which will be established in Section 4, the other generation principles are proved in this section.

Since the subtyping system is separated from the systems for the other judgements, it is immediate to obtain a simple form of generation of subtyping.

Proposition 3.1 (Generation for subtyping)

$$\begin{array}{rcl} \Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B' & \Rightarrow & \Gamma \vdash A' \leq A \ \wedge \ \Gamma, x : A' \vdash B \leq B' \\ \Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A'.B' & \Rightarrow & A' =_\beta A \ \wedge \ \Gamma, x : A' \vdash B \leq B' \end{array}$$

The generation for kinding tells us what information we can infer from a kinding judgement about a kind.

Proposition 3.2 (Generation for kinding)

$$\begin{array}{rcl} \Gamma \vdash \alpha : K & \Rightarrow & K =_\beta Kind_\Gamma(\alpha) \\ \Gamma \vdash \pi x{:}A.B : K & \Rightarrow & K = \star \ \wedge \ \Gamma, x : A \vdash B : \star \\ \Gamma \vdash \Lambda x{:}A.B : K & \Rightarrow & \exists K' \ s.t. \ K =_\beta \Pi x{:}A.K' \ \wedge \ \Gamma, x : A \vdash B : K' \\ \Gamma \vdash AM : K & \Rightarrow & \exists B, K' \ s.t. \ \Gamma \vdash A : \Pi x{:}B.K' \ \wedge \ \Gamma \vdash M : B \ \wedge \ K'[x := M] =_\beta K \end{array}$$

Proof. By an analyse of the kinding derivation. □

The first context property says that the type in a typing assumption of a context is well formed. The second asserts that if a judgement is derivable then the context of the judgement is well-formed.

Proposition 3.3 (Context properties)

1. $\Gamma_1, x : A, \Gamma_2 \vdash J \Rightarrow \Gamma_1 \vdash A : \star$, *where* $\Gamma_1 \vdash A : \star$ *has a smaller proof than* $\Gamma_1, x : A, \Gamma_2 \vdash J$.
2. *Suppose* $\Gamma \vdash J$. *Then for every prefix* Γ' *of* Γ, $\Gamma' \vdash \star$

Uniqueness of kinding can be obtained by observing the fact that any kind is of the form $\Pi x_1{:}A_1 \,..\, \Pi x_n{:}A_n.\star$ and $\Pi x_1{:}A_1 \,..\, \Pi x_n{:}A_n.\star =_\beta \Pi x_1{:}A'_1 \,..\, \Pi x_m{:}A'_m.\star$ iff $n = m, A_i =_\beta A'_i, i = 1..n$.

Proposition 3.4 (Unicity of kinds)

$$\Gamma \vdash A : K \ \wedge \ \Gamma \vdash A : K' \ \Rightarrow \ K =_\beta K'$$

The following three results concerning bound changes will be used in the proof of admissibility of transitivity.

Proposition 3.5 (Bound change for subtyping)

$$\Gamma_1, x : A, \Gamma_2 \vdash B \leq C \;\Rightarrow \Gamma_1, x : A', \Gamma_2 \vdash B \leq C$$

and there exists a derivation of the consequent which is not longer than $\Gamma_1, x : A, \Gamma_2 \vdash B \leq C$.

Proof. By induction on the derivation of $\Gamma_1, x : A, \Gamma_2 \vdash B \leq C$. □

At this moment, we can not prove the bound narrowing for subtyping with bounded type variable:

$$\Gamma_1, \alpha \leq A : K, \Gamma_2 \vdash B \leq C \;\Rightarrow \Gamma_1, \alpha \leq A' : K, \Gamma_2 \vdash B \leq C$$

since the case of S-ApT requiring transitivity which we have not proved yet.

Bound β-equivalence can be viewed as the simplified form of bound narrowing in $\lambda P_{\underline{\leq}}$. Since we have not proved the reflexivity for subtyping: $A =_\beta B \Rightarrow \Gamma \vdash A \leq B$, we need both bound β-equivalence and bound narrowing.

Proposition 3.6 (Bound β-equivalence) *Suppose* $A =_\beta A'$ *and* $\Gamma_1 \vdash A, A' : K$

$$\begin{array}{rcl} \Gamma_1, x : A, \Gamma_2 \vdash J & \Rightarrow & \Gamma_1, x : A', \Gamma_2 \vdash J \\ \Gamma_1, \alpha \leq A : K, \Gamma_2 \vdash J & \Rightarrow & \Gamma_1, \alpha \leq A' : K, \Gamma_2 \vdash J \end{array}$$

Proof. By induction on the derivation of the judgement. □

Proposition 3.7 (Bound narrowing) *Suppose* $\Gamma_1 \vdash A' \leq A$ *and* $\Gamma_1 \vdash A, A' : K$

$$\Gamma_1, x : A, \Gamma_2 \vdash J \quad \Rightarrow \quad \Gamma_1, x : A', \Gamma_2 \vdash J$$

Proof. By induction on the derivation of the judgement. In the case of subsumption, use bound change for subtyping (Proposition 3.5). □

Substitution property is essential in the proof of subject reduction. First, we prove substitution for subtyping.

Proposition 3.8 (Substitution for subtyping)

$$\Gamma_1, x : A, \Gamma_2 \vdash B \leq C \;\wedge\; \Gamma_1, x : A, \Gamma_2 \vdash \star \quad \Rightarrow \quad \Gamma_1, \Gamma_2[x := M] \vdash B[x := M] \leq C[x := M]$$

Proof. By induction on the derivation of $\Gamma_1, x : A, \Gamma_2 \vdash B \leq C$.

Case. (S-ApT). Suppose that the derivation ends by

$$\frac{\Gamma_1, x : A, \Gamma_2 \vdash (\Gamma_1, x : A, \Gamma_2)(\alpha) M_1 .. M_n \leq C}{\Gamma_1, x : A, \Gamma_2 \vdash \alpha M_1 .. M_n \leq C}\ \text{S-ApT}$$

It follows from $\Gamma_1, x : A, \Gamma_2 \vdash \star$ that $x \notin Fv(\Gamma_1)$, so $((\Gamma_1, x : A, \Gamma_2)(\alpha))[x := M] = (\Gamma_1, \Gamma_2[x := M])(\alpha)$. We infer as follows

$$\begin{array}{lll} & \Gamma_1, x : A, \Gamma_2 \vdash (\Gamma_1, x : A, \Gamma_2)(\alpha) M_1 .. M_n \leq C & \\ \Rightarrow & \Gamma_1, \Gamma_2[x := M] \vdash ((\Gamma_1, x : A, \Gamma_2)(\alpha) M_1 .. M_n)[x := M] \leq C[x := M] & IH \\ \Rightarrow & \Gamma_1, \Gamma_2[x := M] \vdash (\Gamma_1, \Gamma_2[x := M])(\alpha) M_1[x := M] .. M_n[x := M] \leq C[x := M] & \\ \Rightarrow & \Gamma_1, \Gamma_2[x := M] \vdash \alpha M_1[x := M] .. M_n[x := M] \leq C[x := M] & S - ApT \\ \equiv & \Gamma_1, \Gamma_2[x := M] \vdash (\alpha M_1 .. M_n)[x := M] \leq C[x := M] & \end{array}$$

Case. (S-ApSL). Suppose that the derivation ends by

$$\frac{\Gamma_1, x:A, \Gamma_2 \vdash F[y := M_1]M_2..M_n \leq C}{\Gamma_1, x:A, \Gamma_2 \vdash (\Lambda y{:}E.F)M_1..M_n \leq C}\ \text{S-ApSL}$$

Without loss of generality, we can assume that $y \notin Fv(M)$. By induction assumption, we have

$$\begin{array}{lll}
& \Gamma_1, \Gamma_2[x := M] \vdash (F[y := M_1]M_2..M_n)[x := M] \leq C[x := M] & \\
\Leftrightarrow & \Gamma_1, \Gamma_2[x := M] \vdash (F[x := M])[y := M_1[x := M]]M_2[x := M]..M_n[x := M] \leq C[x := M] & y \notin Fv(M) \\
\Rightarrow & \Gamma_1, \Gamma_2[x := M] \vdash (\Lambda y{:}E[x := M].F[x := M])M_1[x := M]..M_n[x := M]) \leq C[x := M] & \text{S-ApSL} \\
\Leftrightarrow & \Gamma_1, \Gamma_2[x := M] \vdash ((\Lambda y{:}E.F)M_1..M_n)[x := M] \leq C[x := M] &
\end{array}$$

Other cases are similar. □

The substitution property for typing, kinding and context formation follows.

Proposition 3.9 (Substitution)

$$\Gamma_1, x:A, \Gamma_2 \vdash J \ \wedge\ \Gamma_1 \vdash M : A \ \Rightarrow\ \Gamma_1, \Gamma_2[x := M] \vdash J[x := M]$$

Proof. By induction on the derivation of $\Gamma_1, x:A, \Gamma_2 \vdash J$. □

Agreement of judgements is also frequently used in the proofs.

Proposition 3.10 (Agreement of judgements)

$$\begin{array}{lll}
\Gamma \vdash A : K & \Rightarrow & \Gamma \vdash K \\
\Gamma \vdash M : A & \Rightarrow & \Gamma \vdash A : \star
\end{array}$$

Proof. By induction on the derivation. Using substitution in the case of (T-app) for the second judgement. □

Proof of confluence can be adapted from [HHP93]. Confluence of β reduction holds for pre-expressions.

Proposition 3.11 (Church-Rosser property) *Suppose U, U', U'' are pre-expressions. If $U \twoheadrightarrow_\beta U'$ and $U \twoheadrightarrow_\beta U''$, then there exists a pre-expression V such that $U' \twoheadrightarrow_\beta V$ and $U'' \twoheadrightarrow_\beta V$.*

Corollary 3.12 (Church-Rosser for β-equivalence) *Suppose U, U' are pre-expressions and $U =_\beta U'$, then there exists a pre-expression V such that $U \twoheadrightarrow_\beta V$ and $U' \twoheadrightarrow_\beta V$.*

In order to prove the admissibilities of reflexivity and transitivity, we need an induction measure based on the β_2 head reduction β_{2h}:

$$(\Lambda x : A.B)N_1..N_n \ \rightarrow_{\beta_{2h}}\ B[x := N_1]N_2..N_n$$

It is evident that β_{2h}-reduction is confluent since there are no more than one redex in one type. It also has subject reduction property and strong normalization property.

Lemma 3.13 (β_2 head subject reduction)

$$\Gamma \vdash (\Lambda x : A.B)M_1..M_n : K \Rightarrow \Gamma \vdash B[x := M_1]M_2..M_n : K$$

Proposition 3.14 (β_{2h} strongly normalizing) *If $\Gamma \vdash A : K$, then A is strongly β_{2h}-normalizing.*

The proof is from the observation that any β_{2h} reducible type or type family (like A in the proposition) must be of the form $(\Lambda x_1{:}A_1..\Lambda x_n{:}A_n.\alpha M_1..M_k)N_1..N_h$ for $n \geq 1, k \geq 0, h \geq 0$, whose reduction should be in less than n steps. Note that, one can not have something like $(\Lambda x{:}A.xM)N$ because of the kinding.

4 Admissible Rules in Subtyping System

In this section, we study the admissibilities of reflexivity, transitivity and the rule for subtyping type family application, namely

$$\frac{\Gamma \vdash A, B : K \quad A =_\beta B}{\Gamma \vdash A \leq B} \qquad \frac{\Gamma \vdash A, B, C : K \quad \Gamma \vdash A \leq B \quad \Gamma \vdash B \leq C}{\Gamma \vdash A \leq C} \qquad \frac{\Gamma \vdash AM, BM : K \quad \Gamma \vdash A \leq B}{\Gamma \vdash AM \leq BM}$$

The kinding premises are necessary for these rules. To prove the admissibility of reflexivity rule, one need to analyse according to the forms of A, B. Without well-kindness requirement, it is possible to have an instance of reflexivity as

$$\frac{(\pi x : C.D)M =_\beta (\pi x : C'.D')M'}{(\pi x : C.D)M \leq (\pi x : C'.D')M'}$$

which is neither desirable nor provable by the subtyping rules.

The proof of transitivity will use the admissiblility of reflexivity. Therefore, kinding premises are also required. For the rule of subtyping family, we do not want cases like $(\pi x : C.D)M \leq (\pi x : C'.D')M$, thus we need the well-kindness for types in the subtyping judgement.

Lemma 4.1 (Well-kindness of subtyping) *Suppose that* $\Gamma \vdash A \leq B$ *and* $\Gamma \vdash A, B : K$. *Then for any subtyping judgement* $\Gamma \vdash C \leq D$ *in the derivation of* $\Gamma \vdash A \leq B$, *there exists a kind* K' *such that* $\Gamma \vdash C, D : K'$.

Proof. By inspecting the kinding rules. □

4.1 Reflexivity

In many subtyping systems, e.g., $F_{\leq}$ [CG92], $Co^{\vdash}$[LMS95], the reflexivity is trivial

$$\frac{A = A}{\Gamma \vdash A \leq A}\ refl$$

But for systems with a reduction relation defined on types, the reflexivity becomes complicated. It says that types equivalent in reduction have the subtyping relation.

The proof of admissibility of a rule R is essentially a transformation procedure from a derivation using R to a derivation without R. To show such process terminates, one needs an induction measure. We borrow an idea from [AC96] by using a measure based on lexicographic combination of maximal steps of some reduction and size of type. Here, the reduction is β_{2h}, which has strong normalization property (Prop. 3.14), so such measure is well-defined.[1]

Proposition 4.2 (Admissibility of reflexivity)

$$A =_\beta B \ \wedge\ \Gamma \vdash A, B : K \quad \Rightarrow \quad \Gamma \vdash A \leq B$$

Proof. Since A, B are well kinded, they can only take the following forms

$$\begin{array}{ll} \alpha M_1..M_n & n \geq 0 \\ \Lambda x{:}C.D & \\ \pi x{:}C.D & \\ (\Lambda x : C.D)M_1..M_n & n \geq 1 \end{array}$$

[1] In [AC96], the measure is based on β_2-reduction and an extension of types by a type operator *plus*. Ours is simpler.

Define an induction measure $weight(A, B)$ on the pair of A, B:

$$weight(A, B) =< MaxRed2h(A) + MaxRed2h(B), Size(A) + Size(B) >$$

where $MaxRed2h(A)$ is the maximum number of β_{2h} reduction steps from A:

$$MaxRed2h(A) = max\{n \mid A \rightarrow^{n}_{\beta_{2h}} A^{\beta_{2h}}\}$$

and $Size(A)$ is the number of symbols in A. Note that $weight(A, B)$ is well defined for kindable types.

We proceed by induction on $weight(A, B)$.

Case1 $(\alpha M_1..M_n =_\beta \alpha M'_1..M'_n)$. By S-ApR.

Case2 $(\alpha M_1..M_n =_\beta \Lambda x{:}C.D)$. By the Church-Rosser property (Proposition 3.12), if such conversion exists, then there is a term N such that $\alpha M \rightarrow^*_R N$ and $\Lambda x{:}C.D \rightarrow^*_R N$. That's impossible.

Case3 $(\alpha M_1..M_n =_\beta \pi x{:}C.D)$. Impossible by the same reason.

Case4 $(\alpha M_1..M_n =_\beta (\Lambda x : C.D)N_1..N_k)$ for $k \geq 1$. By the assumption,

$$\Gamma \vdash (\Lambda x : C.D)N_1..N_k : K \ \Rightarrow \Gamma \vdash D[x := N_1]N_2..N_k : K \qquad prop.\ 3.13$$

So

$$\begin{array}{lll}
 & \Gamma \vdash \alpha M_1..M_n, D[x := N_1]N_2..N_k : K \ \wedge & \\
 & \alpha M_1..M_n =_\beta (\Lambda x : C.D)N_1..N_k =_\beta D[x := N_1]N_2..N_k \ \wedge & \\
 & weight(\alpha M_1..M_n, D[x := N_1]N_2..N_k) \leq weight(\alpha M_1..M_n, (\Lambda x : C.D)N_1..N_k) & \\
\Rightarrow & \Gamma \vdash \alpha M_1..M_n \leq D[x := N_1]N_2..N_k & IH \\
\Rightarrow & \Gamma \vdash \alpha M_1..M_n \leq (\Lambda x : C.D)N_1..N_k & \text{S-ApSR}
\end{array}$$

Case5 $(\Lambda x{:}C.D =_\beta \Lambda x{:}C'.D')$. By the Church-Rosser property (Proposition 3.12), $C =_\beta C', D =_\beta D'$. Thus, if we can prove that $\Gamma \vdash C, C' : \star$ and $\Gamma, x : C \vdash D, D' : K_1$ for some kind K_1, then the result follows from the induction assumption and the rule (S-Λ). Now we prove these kinding judgements.

$$\begin{array}{lll}
 & \Gamma \vdash \Lambda x{:}C.D, \Lambda x{:}C'.D' : K & \\
\Rightarrow & K =_\beta \Pi x{:}C.K_1 =_\beta \Pi x{:}C'.K_2 \ \wedge \ \Gamma, x : C \vdash D : K_1 \ \wedge \ \Gamma, x : C' \vdash D' : K_2 & prop.\ 3.2 \\
\Rightarrow & K_1 =_\beta K_2 \ \wedge \ \Gamma \vdash C, C' : \star & prop.\ 3.12,\ 3.3
\end{array}$$

$$\begin{array}{lll}
\Gamma, x : C' \vdash D' : K_2 \ \wedge \ C =_\beta C' \ \wedge \ \Gamma \vdash C, C' : \star & \Rightarrow & \Gamma, x : C \vdash D' : K_2 \quad prop.\ 3.6 \\
\Gamma, x : C \vdash D' : K_2 \ \wedge \ K_1 =_\beta K_2 & \Rightarrow & \Gamma, x : C \vdash D' : K_1 \quad \text{K-conv}
\end{array}$$

Thus, we have obtained the desired result $\Gamma, x : C \vdash D', D : K_1$.

Other cases are similar.

□

4.2 Transitivity

In this subsection, we show that the subtyping relation in $\lambda\Pi_\leq$ is transitive for kindable types. That is, the rule:

$$\frac{\Gamma \vdash A, B, C : K \quad \Gamma \vdash A \leq C \quad \Gamma \vdash C \leq B}{\Gamma \vdash A \leq B}\ trans$$

is admissible.

Observe that the proof of admissibility of transitivity can be obtained from a transitivity-elimination process for the subtyping system extended with the above transitivity rule (we denote the system by $\lambda\Pi^t_{\leq}$). The judgement in the extended system will be denoted by $\vdash_t$.

As refered in the beginning of this section, kinding premises are required for the transitivity rule. Here we make a more detailed analyseon this problem.

Consider a transitivity application

$$\dfrac{\dfrac{M_1 =_\beta M'_1 \quad \cdots \quad M_n =_\beta M'_n}{\Gamma \vdash \alpha M_1..M_n \leq \alpha M'_1..M'_n}\ \text{S-ApR} \quad \dfrac{\vdots}{\Gamma \vdash \Gamma(\alpha)M'_1..M'_n \leq C}\ \text{S-ApT}}{\Gamma \vdash_t \alpha M_1..M_n \leq C}\ trans$$

we would like to transform it to

$$\dfrac{\dfrac{\vdots}{\Gamma \vdash \Gamma(\alpha)M_1..M_n \leq \Gamma(\alpha)M'_1..M'_n} \quad \dfrac{\vdots}{\Gamma \vdash \Gamma(\alpha)M'_1..M'_n \leq C}\ trans}{\dfrac{\Gamma \vdash_t \Gamma(\alpha)M_1..M_n \leq C}{\Gamma \vdash_t \alpha M_1..M_n \leq C}\ \text{S-AppT}}$$

where we need the reflexivity to show that the the judgement $\Gamma \vdash \Gamma(\alpha)M_1..M_n \leq \Gamma(\alpha)M'_1..M'_n$ could be proved in $\lambda\Pi_{\leq}$. Hence, the pair of types $\Gamma(\alpha)M_1..M_n, \Gamma(\alpha)M'_1..M'_n$ should be well-kinded to permit the application of reflexivity.

This example also raises second problem, that is, the measure for induction in the transtivity elimination proof. The usual approach to prove the transitivity elimination is to transform derivation ended by a transitivity application to a new one in which depths of transitivity subderivations are shorter. But in our previous example, this can not be ensured because the derivation of $\Gamma \vdash \Gamma(\alpha)M_1..M_n \leq \Gamma(\alpha)M'_1..M'_n$ may be longer than that of $\Gamma \vdash \alpha M_1..M_n \leq \alpha M'_1..M'_n$.

Notice that from $\alpha M_1..M_n$ to $\Gamma(\alpha)M_1..M_n$ is one step Γ-reduction, that is, a reduction by replacing the type variable by its upper bound. So we can consider using $\beta_{2h}\Gamma$ reduction (combination of β_{2h} and Γ reduction) to construct an induction measure. First, we have the property that $\beta_{2h}\Gamma$ reduction is strongly normalizing.

Lemma 4.3 ($\beta_{2h}\Gamma$ strongly normalizing) *If $\Gamma \vdash A : K$, then A is strongly $\beta_{2h}\Gamma$-normalizing.*

The proof can be adapted from the one presented in section 6.

Now, we can compute the maximal steps of $\beta_{2h}\Gamma$ normalization and define an appropriate measure,

$$weight(A, B) =< MaxRed2hg(A) + MaxRed2hg(B), Size(A) + Size(B) >$$

where $MaxRed2hg(A)$ is the maximum number of $\beta_{2h}\Gamma$ reduction steps from A:

$$MaxRed2hg(A) = max\{n \mid A \rightarrow^n_{\beta_{2h}\Gamma} A^{\beta_{2h}}\}$$

Now we can show the transitivity elimination for $\lambda\Pi^t_{\leq}$: a derivation for a $\vdash_t$ judgement can be replaced by a derivation in $\lambda\Pi_{\leq}$.

Proposition 4.4 (Transitivity elimination in $\lambda\Pi^t_{\leq}$)

$$\Gamma \vdash_t A \leq B \Rightarrow \Gamma \vdash A \leq B$$

Proof. Let us consider derivations in the extended system with one application of the rule of transitivity in the end. If we can prove that it is equal to a transitivity-free derivation, we may then eliminate transitivities from arbitrary derivations one by one, beginning with uppermost application of transitivity.

Consider then the first application of transitivity in a derivation

$$\frac{\Gamma \vdash U, V, W : K \quad \Gamma \vdash U \leq V \quad \Gamma \vdash V \leq W}{\Gamma \vdash_t U \leq W}\ trans$$

The derivations of $\Gamma \vdash U \leq V$ and $\Gamma \vdash V \leq W$ are transitivity-free, i.e. they are derivations in $\lambda\Pi_{\leq}$, whereas $\Gamma \vdash_t U \leq W$ is a $\lambda\Pi^t_{\leq}$ judgement.

Induction on $weight(U, W)$. We proceed by case analysis of the last pair of rules used to derive $\Gamma \vdash_t U \leq W$. We show that the derivation can be transformed to one in which each transivity application is smaller than the original one by the measure $weight$. Thus, by induction, those subderivations can be transformed to transitivity-free derivations.

Case. (S-π, S-π). The derivation must end by

$$\frac{KJ \qquad \dfrac{\Gamma \vdash A_2 \leq A_1 \quad \Gamma, x : A_2 \vdash B_1 \leq B_2}{\Gamma \vdash \pi x{:}A_1.B_1 \leq \pi x{:}A_2.B_2} \qquad \dfrac{\Gamma \vdash A_3 \leq A_2 \quad \Gamma, x : A_3 \vdash B_2 \leq B_3}{\Gamma \vdash \pi x{:}A_2.B_2 \leq \pi x{:}A_3.B_3}}{\Gamma \vdash_t \pi x{:}A_1.B_1 \leq \pi x{:}A_3.B_3}\ trans$$

where KJ is the kinding judgement $\Gamma \vdash \pi x{:}A_1.B_1, \pi x{:}A_2.B_2, \pi x{:}A_3.B_3 : \star$.

We infer a few deducible kindabilities and subtyping relation.

$$\Gamma \vdash \pi x{:}A_1.B_1, \pi x{:}A_2.B_2, \pi x{:}A_3.B_3 : \star \ \Rightarrow$$
$$\Gamma \vdash A_1, A_2, A_3 : \star \ \wedge\ \Gamma, x : A_1 \vdash B_1 : \star \wedge \Gamma, x : A_2 \vdash B_2 : \star \ \wedge\ \Gamma, x : A_3 \vdash B_3 : \star \quad prop.\ 3.2$$

$$\begin{array}{lcll}
\Gamma \vdash A_3, A_2 : \star \ \wedge\ \Gamma \vdash A_3 \leq A_2 \ \wedge\ \Gamma, x : A_2 \vdash B_1 \leq B_2 & \Rightarrow & \Gamma, x : A_3 \vdash B_1 \leq B_2 & prop.\ 3.5 \\
\Gamma \vdash A_1, A_2, A_3 : \star \ \wedge\ \Gamma \vdash A_3 \leq A_2 \ \wedge\ \Gamma \vdash A_2 \leq A_1 & \Rightarrow & \Gamma \vdash A_3 \leq A_1 & IH \\
\Gamma \vdash A_3, A_2 : \star \ \wedge\ \Gamma \vdash A_3 \leq A_2 \ \wedge\ \Gamma, x : A_2 \vdash B_2 : \star & \Rightarrow & \Gamma, x : A_3 \vdash B_2 : \star & prop.\ 3.7 \\
\Gamma \vdash A_3, A_2 : \star \ \wedge\ \Gamma \vdash A_3 \leq A_1 \ \wedge\ \Gamma, x : A_1 \vdash B_1 : \star & \Rightarrow & \Gamma, x : A_3 \vdash B_1 : \star & prop.\ 3.7
\end{array}$$

In conclusion, we have obtained $\Gamma, x : A_3 \vdash B_1, B_2, B_3 : \star$ and $\Gamma, x : A_3 \vdash B_1 \leq B_2$.

Therefore the above derivation with transitivity could be transformed to

$$\frac{\dfrac{JK1 \quad \Gamma \vdash A_3 \leq A_2 \quad \Gamma \vdash A_2 \leq A_1}{\Gamma \vdash_t A_3 \leq A_1}\ trans \qquad \dfrac{JK2 \quad \Gamma, x : A_3 \vdash B_1 \leq B_2 \quad \Gamma, x : A_3 \vdash B_2 \leq B_3}{\Gamma, x : A_3 \vdash_t B_1 \leq B_3}\ trans}{\Gamma \vdash_t \pi x{:}A_1.B_1 \leq \pi x : A_3.B_3}$$

where $JK1 \equiv \Gamma \vdash A_1, A_2, A_3 : \star$ and $JK2 = \Gamma, x : A_3 \vdash B_1, B_2, B_3 : \star$. The induction measure reduces because of the decrease of sizes of types.

Case. (S-Λ, S-Λ) Similar.

Case. (S-ApR, S-ApR) by transitivity of β-conversion.

Case. (S-ApSL, _) The derivation must end by

$$\frac{\dfrac{\Gamma \vdash (B[x := M_1])M_2..M_n \leq C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq C} \qquad \Gamma \vdash C \leq D \quad JK}{\Gamma \vdash_t (\Lambda x{:}A.B)M_1..M_n \leq D}\ trans$$

where $JK \equiv \Gamma \vdash (\Lambda x{:}A.B)M_1..M_n, D, C : K$. By the β_{2h} subject reduction lemma (lemma 3.13), $(B[x := M_1])M_2..M_n$ is well-kinded in the context Γ. So the above derivation could be transformed to

$$\dfrac{\dfrac{JK \quad \Gamma \vdash (B[x := M_1])M_2..M_n \leq C \quad \Gamma \vdash C \leq D}{\Gamma \vdash_t (B[x := M_1])M_2..M_n \leq D}\ trans}{\Gamma \vdash_t (\Lambda x{:}A.B)M_1..M_n \leq D}$$

where $JK' \equiv \Gamma \vdash (B[x := M_1])M_2..M_n, D, C : K$. The induction measure reduces because of the β_{2h}-reduction.

Case. (_, S-ApSR), (S-ApT, _). Similar.

Case. (S-ApR, S-ApT) The derivation must end by

$$\dfrac{JK \quad \dfrac{M_1 =_\beta M_1' \quad \cdots \quad M_n =_\beta M_n'}{\Gamma \vdash \alpha M_1..M_n \leq \alpha M_1'..M_n'}\ \text{S-ApR} \quad \dfrac{\overline{\Gamma \vdash \Gamma(\alpha)M_1'..M_n' \leq C}}{\Gamma \vdash \alpha M_1'..M_n' \leq C}\ \text{S-ApT}}{\Gamma \vdash_t \alpha M_1..M_n \leq C}\ trans$$

where $JK \equiv \Gamma \vdash \alpha M_1..M_n, \alpha M_1'..M_n', C : K$. From the kinding assumption in the transitivity, it follows that

$$\Gamma \vdash \Gamma(\alpha)M_1..M_n, \Gamma(\alpha)M_1'..M_n', C : K$$

By the reflexivity of subtyping (Proposition 4.2), the judgement $\Gamma \vdash_t \Gamma(\alpha)M_1..M_n \leq \Gamma(\alpha)M_1'..M_n'$ is derivable. Therefore, we can have a derivation ending by

$$\dfrac{\dfrac{JK \quad \Gamma \vdash \Gamma(\alpha)M_1..M_n \leq \Gamma(\alpha)M_1'..M_n' \quad \Gamma \vdash \Gamma(\alpha)M_1'..M_n' \leq C}{\Gamma \vdash_t \Gamma(\alpha)M_1..M_n \leq C}\ trans}{\Gamma \vdash_t \alpha M_1..M_n \leq C}\ \text{S-ApT}$$

where $JK \equiv \Gamma \vdash \Gamma(\alpha)M_1..M_n, \Gamma(\alpha)M_1'..M_n', C : K$. The induction measure reduces because of the Γ-reduction.

Case. (S-ApSR, S-ApSL). The derivation must end by

$$\dfrac{JK \quad \dfrac{\Gamma \vdash C \leq B[x := M_1]M_2..M_n}{\Gamma \vdash C \leq (\Lambda x{:}A.B)M_1..M_n}\ \text{S-ApSR} \quad \dfrac{\Gamma \vdash B[x := M_1]M_2..M_n \leq C'}{\Gamma \vdash (\Lambda x : A.B)M_1..M_n \leq C'}\ \text{S-ApSL}}{\Gamma \vdash_t C \leq C'}\ trans$$

where $JK \equiv \Gamma \vdash C, (\Lambda x{:}A.B)M_1..M_n, B[x := M_1]M_2..M_n : K$. The derivation could be transformed to

$$\dfrac{JK \quad \Gamma \vdash C \leq B[x := M_1]M_2..M_n \quad \Gamma \vdash B[x := M_1]M_2..M_n \leq C'}{\Gamma \vdash_t C \leq C'}\ trans$$

where $JK \equiv \Gamma \vdash C, B[x := M_1]M_2..M_n, C' : K$

Other combinations of rules are impossible. □

Corollary 4.5 (Admissibility of transitivity in subtyping)

$$\Gamma \vdash A, B, C : K \ \wedge\ \Gamma \vdash A \leq B \ \wedge\ \Gamma \vdash B \leq C \ \Rightarrow\ \Gamma \vdash A \leq C$$

4.3 Subtyping family application

Now we want to show the admissibility of the subtyping family application rule

$$\frac{\Gamma \vdash A, B, C : K \quad \Gamma \vdash AM, BM : K \quad \Gamma \vdash A \leq B}{\Gamma \vdash AM \leq BM}$$

This property and the reflexivity are specific to subtyping in dependent types. As for reflexivity property, the well kindness for types in subtyping judgement is required. First, we prove a lemma.

Lemma 4.6 $\Gamma \vdash \alpha M_1..M_n : K \ \wedge \ \alpha$ *bound in* $\Gamma \ \Rightarrow \ \Gamma \vdash \Gamma(\alpha)M_1..M_n : K$

Proof. By the observation that there are $A_1, .., A_n$ and K' such that $\alpha \leq \Gamma(\alpha) : \Pi x_1{:}A_1..\Pi x_n{:}A_n.K' \in \Gamma$ where $K'[\overline{x} := \overline{M}] = K$. □

The main result of this subsection can be established.

Proposition 4.7 (Subtyping family application)

$$\Gamma \vdash AM, BM : K \ \wedge \ \Gamma \vdash A \leq B \ \Rightarrow \ \Gamma \vdash AM \leq BM$$

Proof. By induction on the derivation of $\Gamma \vdash A \leq B$.
Case (S-π). Impossible since AM, BM can not be well-kinded.

Case (S-Λ). Suppose that the derivation ends with

$$\frac{A' =_\beta A \quad \Gamma, x : A \vdash B \leq B'}{\Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A'.B'} \text{ S-}\Lambda$$

By the substitution for subtyping property (Proposition 3.8), $\Gamma, x : A \vdash B \leq B' \ \Rightarrow \ \Gamma \vdash B[x := M] \leq B'[x := M]$. So we have a derivation ending by

$$\frac{\dfrac{\Gamma \vdash B[x := M] \leq B'[x := M]}{\Gamma \vdash B[x := M] \leq (\Lambda x{:}A'.B')M} \text{ S-ApSR}}{\Gamma \vdash (\Lambda x{:}A.B)M \leq (\Lambda x{:}A'.B')M} \text{ S-ApSL}$$

Case (S-ApR). Immediate.

Case (S-ApT). By Lemma 4.6

$$\Gamma \vdash \alpha M_1..M_nM : K \quad \Rightarrow \quad \Gamma \vdash \Gamma(\alpha)M_1..M_nM : K$$

The result follows from induction assumption.

Case (S-ApSL). Suppose that the derivation ends with

$$\frac{\Gamma \vdash B[x := M_1]M_2..M_n \leq C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq C} \text{ S-ApSL}$$

Starting from the assumption, we infer as follows

	$\Gamma \vdash ((\Lambda x{:}A.B)M_1..M_n)M : K$	
$\Rightarrow$	$\exists D, K' \ s.t. \ \Gamma \vdash (\Lambda x{:}A.B)M_1..M_n : \Pi y{:}D.K' \ \wedge \ K = K'[x := M] \ \wedge \ \Gamma \vdash M : D$	*prop.* 3.2
$\Rightarrow$	$\Gamma \vdash B[x := M_1]M_2..M_n : \Pi y{:}D.K'$	*subject reduction*
$\Rightarrow$	$\Gamma \vdash B[x := M_1]M_2..M_nM : K'[x := M]$	K-app

By the β_{2h} subject reduction (lemma 3.13), $(B[x := M_1])M_2..M_nM$ is well-kinded in the context Γ. Therefore,

$$\begin{array}{lll} & \Gamma \vdash B[x := M_1]M_2..M_nM, CM : K \ \wedge \ \Gamma \vdash B[x := M_1]M_2..M_n \leq C & \\ \Rightarrow & \Gamma \vdash B[x := M_1]M_2..M_nM \leq CM & IH \\ \Rightarrow & \Gamma \vdash (\Lambda x{:}A.B)M_1..M_nM \leq CM & \text{S-ApSL} \end{array}$$

Case (S-ApSR). Similar. □

5 Subject Reduction

In this section, we prove the generation for typing property and the subject reduction. Proofs of both results need the transitivity of subtyping, which is used to show the following fact: any typing proof can be transformed to a proof in which there is no consecutive application of subsumption.

The generation for typing tells us what information we can infer from a typing judgement about a type.

Proposition 5.1 (Generation for typing)

$$\begin{array}{rcl} \Gamma \vdash x : C & \Rightarrow & \Gamma \vdash \Gamma(x) \leq C \\ \Gamma \vdash \lambda x{:}A.M : C & \Rightarrow & \exists B \ s.t. \ \Gamma, x : A \vdash M : B \ \wedge \ \Gamma \vdash \pi x{:}A.B \leq C \\ \Gamma \vdash MN : C & \Rightarrow & \exists A, B \ s.t. \ \Gamma \vdash M : \pi x{:}A.B \ \wedge \ \Gamma \vdash N : A \ \wedge \ \Gamma \vdash B[x := N] \leq C \end{array}$$

Proof. Without loss of generality, it is assumed that there are no consecutive applications of assumption. The result follows from induction on typing derivation. □

Subject reduction is one of the main concerns in the study of subtyping dependent types. Since subtyping is separated from other judgements, we give here an alternative proof than that in [AC96].

Proposition 5.2 (Subject reduction)

$$\Gamma \vdash J \ \wedge \ J \rightarrow_\beta^* J' \ \Rightarrow \ \Gamma \vdash J'$$

Proof. It is enough to prove the one step case, which follows by induction on the derivation of the judgement $\Gamma \vdash J$. Without loss of generality, we assume that there are no two consecutive applications of assumption.

The main case is $\Gamma \vdash (\lambda x{:}A.M)N : C$ and $(\lambda x{:}A.M)N \rightarrow_\beta M[x := N]$, we need to prove that $\Gamma \vdash M[x := N] : C$.

If $\Gamma \vdash (\lambda x{:}A.M)N : C$ is derived by T-app, then there are A' and B such that $\Gamma \vdash N : A'$, $\Gamma \vdash \lambda x{:}A.M : \pi x{:}A'.B$ and $C = B[x := N]$. Otherwise the derivation must end by

$$\dfrac{\dfrac{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A'.B \quad \Gamma \vdash N : A'}{\Gamma \vdash (\lambda x{:}A.M)N : B[x := N]}\ \text{T-app} \qquad \Gamma \vdash B[x := N] \leq C \quad \Gamma \vdash B[x := N], C : \star}{\Gamma \vdash (\lambda x{:}A.M)N : C}\ \text{T-sub}$$

In any of these two cases, if $\Gamma \vdash \lambda x{:}A.M : \pi x{:}A'.B$ is derived by (T-λ), then $A' = A$ and $\Gamma, x : A' \vdash M : B$, otherwise it must be derived by

$$\dfrac{\dfrac{\Gamma, x : A \vdash M : B'}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A.B'} \qquad \dfrac{\Gamma \vdash A' \leq A \quad \Gamma, x : A' \vdash B' \leq B}{\Gamma \vdash \pi x{:}A.B' \leq \pi x{:}A'.B} \qquad \Gamma \vdash \pi x{:}A.B', \pi x{:}A'.B : \star}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A'.B}\ \text{T-sub}$$

By generation for kinding (Proposition 3.2), $\Gamma, x : A' \vdash B, B' : \star$, so we can apply subsumption

$$\frac{\Gamma, x : A' \vdash M : B' \quad \Gamma, x : A' \vdash B' \leq B \quad \Gamma, x : A' \vdash B, B' : \star}{\Gamma, x : A' \vdash M : B} \text{ T-sub}$$

In all cases it follows from the substitution (Proposition 3.9) that $\Gamma \vdash M[x := N] : B[x := N]$. Apply subsumption (when necessary), we get $\Gamma \vdash M[x := N] : C$.

Another delicate case is $\Gamma \vdash \Pi x{:}A.K$ and $A \rightarrow_\beta A'$. In this case, we have a context formation derivation ended with

$$\frac{\Gamma, x : A \vdash K}{\Gamma \vdash \Pi x{:}A.K} \text{ F-}\Pi$$

It follows from the context property (Proposition 3.3) that $\Gamma \vdash A : \star$ and that its proof is less than that of $\Gamma, x : A \vdash K$. Therefore, we can apply the induction assumption and get $\Gamma \vdash A' : \star$. By bound β-equivalence (Proposition 3.6) and the fact that $\Gamma, x : A \vdash K$ and $A =_\beta A'$, we get $\Gamma, x : A' \vdash K$, the result $\Gamma \vdash \Pi x{:}A'.K$ follows.

The other cases are easy. □

The Church-Rosser properties in Section 3 are established on pre-expressions, that is, given U, U', U'' such that $U \twoheadrightarrow_\beta U'$ and $U \twoheadrightarrow_\beta U''$, there exists a pre-expression V such that $U' \twoheadrightarrow_\beta V$ and $U'' \twoheadrightarrow_\beta V$. By the subject redution, if we know U, U', U'' are well-formed, then so does V.

Proposition 5.3 (Church-Rosser property for well-formed expressions)

$$\begin{array}{rcl}
\Gamma \vdash M, M', M'' : A \ \wedge\ M \twoheadrightarrow_\beta M' \ \wedge\ M \twoheadrightarrow_\beta M'' & \Rightarrow & \exists N \ \ \Gamma \vdash N : A \ \wedge\ M' \twoheadrightarrow_\beta N \ \wedge\ M'' \twoheadrightarrow_\beta N \\
\Gamma \vdash A, A', A'' : K \ \wedge\ A \twoheadrightarrow_\beta A' \ \wedge\ A \twoheadrightarrow_\beta A'' & \Rightarrow & \exists B \ \ \Gamma \vdash B : K \ \wedge\ A' \twoheadrightarrow_\beta B \ \wedge\ A'' \twoheadrightarrow_\beta B \\
\Gamma \vdash K, K', K'' \ \wedge\ K \twoheadrightarrow_\beta K' \ \wedge\ K \twoheadrightarrow_\beta K'' & \Rightarrow & \exists H \ \ \Gamma \vdash H \ \wedge\ K' \twoheadrightarrow_\beta H \ \wedge\ K'' \twoheadrightarrow_\beta H
\end{array}$$

6 Strong normalization

Well-formed terms, types and kinds have strong normalization property. The basic idea of the proof is to transform terms, types and kinds to those in $\lambda\Pi$ and show that a reduction in $\lambda\Pi_\leq$ corresponds to one or more steps in $\lambda\Pi$. Note that $\lambda\Pi$ can be obtained from $\lambda\Pi_\leq$ by removing the subtyping rules, the F-subtype rule of bound type variable introduction, and changing the subtyping judgement in the subsumption rule to β-equivalence. We define a transformation function FE_Γ which fully expands bounded type variables in the augument (a preterm, a pretype or a prekind) by their greatest upper bounds in the context:

$$\begin{array}{lcll}
FE_\Gamma(x) & = & x & \\
FE_\Gamma(MN) & = & FE_\Gamma(M)FE_\Gamma(N) & \\
FE_\Gamma(\lambda x{:}A.M) & = & \lambda x{:}FE_\Gamma(A).FE_\Gamma(M) & \\
FE_\Gamma(\alpha) & = & FE_\Gamma(\Gamma(\alpha)) & if\ \alpha \leq \Gamma(\alpha) : K \in \Gamma \\
FE_\Gamma(\alpha) & = & \alpha & if\ \alpha : K \in \Gamma \\
FE_\Gamma(\pi x{:}A.B) & = & \pi x{:}FE_\Gamma(A).FE_\Gamma(B) & \\
FE_\Gamma(\Lambda x{:}A.B) & = & \Lambda x{:}FE_\Gamma(A).FE_\Gamma(B) & \\
FE_\Gamma(AN) & = & FE_\Gamma(A)FE_\Gamma(N) & \\
FE_\Gamma(\star) & = & \star & \\
FE_\Gamma(\Pi x{:}A.K) & = & \Pi x{:}FE_\Gamma(A).FE_\Gamma(K) &
\end{array}$$

For well-formed context Γ, the function FE_Γ is well defined on preterms, pretypes and prekinds.

This function can be extended to context as

$$\begin{array}{lcl} FE(<>) & = & <> \\ FE(\Gamma, x:A) & = & FE(\Gamma), x : FE_\Gamma(A) \\ FE(\Gamma, \alpha : K) & = & FE(\Gamma), \alpha : FE_\Gamma(K) \\ FE(\Gamma, \alpha \leq A : K) & = & FE(\Gamma) \end{array}$$

For well-formed context Γ, the function FE is well-defined.

FE has the following properties:

Lemma 6.1 (Preservation of substitution of FE)

$$\begin{array}{rcl} \Gamma, x:A \vdash N:B \ \wedge \ \Gamma \vdash M:A & \Rightarrow & FE_\Gamma(N[x:=M]) = FE_{\Gamma,x:A}(N)[x := FE_\Gamma(M)] \\ \Gamma, x:A \vdash B:K \ \wedge \ \Gamma \vdash M:A & \Rightarrow & FE_\Gamma(B[x:=M]) = FE_{\Gamma,x:A}(B)[x := FE_\Gamma(M)] \\ \Gamma, x:A \vdash K \ \wedge \ \Gamma \vdash M:A & \Rightarrow & FE_\Gamma(K[x:=M]) = FE_{\Gamma,x:A}(K)[x := FE_\Gamma(M)] \end{array}$$

Proof. By induction on the derivations of $\Gamma, x:A \vdash N:B, \Gamma, x:A \vdash B:K$ *and* $\Gamma, x:A \vdash K$ respectively. □

Since FE only replaces bounded type variables by their upper bounds, so β equality should be preserved.

Lemma 6.2 (Preservation of β-equality by FE) *Suppose that U, V are both well-formed kinds, types or terms, then*

$$U =_\beta V \ \Rightarrow \ FE(U) =_\beta FE(V)$$

Lemma 6.3 (Subtyping transforms to β-equality)

$$\Gamma \vdash A \leq B \ \wedge \ \Gamma \vdash A, B : K \ \Rightarrow \ FE(A) =_\beta FE(B)$$

Proof. Induction on the derivation of $\Gamma \vdash A \leq B$.

Case S-π. Suppose that $A \equiv \pi x{:}C.D$, $B \equiv \pi x{:}C'.D'$ and the judgement $\Gamma \vdash A \leq B$ is derived by the rule S-π. Then

$$\begin{array}{lcll} \Gamma \vdash \pi x{:}C.D, \pi x{:}C'.D' : K & \Rightarrow & \Gamma \vdash C, C' : \star \ \wedge \ \Gamma, x:C \vdash D : \star \wedge \Gamma, x:C' \vdash D' : \star & \textit{prop. } 3.2 \\ \Gamma \vdash \pi x{:}C.D \leq \pi x{:}C'.D' & \Rightarrow & \Gamma \vdash C' \leq C \ \wedge \ \Gamma, x:C' \vdash D \leq D' & \textit{rule } \text{S-}\pi \\ \Gamma \vdash C' \leq C \ \wedge \ \Gamma, x:C \vdash D : \star & \Rightarrow & \Gamma, x:C' \vdash D : \star & \textit{prop. } 3.7 \end{array}$$

$$\begin{array}{lcll} \Gamma, x:C' \vdash D, D' : \star \ \wedge \ \Gamma, x:C' \vdash D \leq D' & \Rightarrow & FE(D) =_\beta FE(D') & IH \\ \Gamma \vdash C' \leq C \ \wedge \ \Gamma \vdash C, C' : \star & \Rightarrow & FE(C) =_\beta FE(C') & IH \\ FE(D) =_\beta FE(D') \ \wedge \ FE(C) =_\beta FE(C') & \Rightarrow & FE(\pi x{:}C.D) =_\beta FE(\pi x{:}C'.D') & \end{array}$$

Case S-Λ. Similar.

Case S-ApR. By Lemma 6.2.

Case S-ApT. Since $\Gamma \vdash \alpha M_1..M_n : K \ \Rightarrow \ \Gamma \vdash \Gamma(\alpha)M_1..M_n : K$, the result follows from the induction assumption and the fact that $FE(\Gamma(\alpha)M_1..M_n) = FE(\alpha M_1..M_n)$.

Case S-ApSL. The result follows from the β_2-subject reduction and induction assumption. □

Lemma 6.4 (Preservation of kinding of FE)

$$\begin{array}{rcl} \Gamma \vdash N:B & \Rightarrow & FE(\Gamma) \vdash_{\lambda\Pi} FE_\Gamma(N) : FE_\Gamma(B) \\ \Gamma \vdash B:K & \Rightarrow & FE(\Gamma) \vdash_{\lambda\Pi} FE_\Gamma(B) : FE_\Gamma(K) \\ \Gamma \vdash K & \Rightarrow & FE(\Gamma) \vdash_{\lambda\Pi} FE_\Gamma(K) \end{array}$$

where $\vdash_{\lambda\Pi}$ denotes judgements in $\lambda\Pi$.

Proof. By induction on the derivations of $\Gamma \vdash N : B, \Gamma \vdash B : K$ *and* $\Gamma \vdash K$ respectively. We analyse the case where the judgement is derived by the subsumption rule, other cases are easy. Suppose that a derivation is ended by:

$$\frac{\Gamma \vdash M : A \quad \Gamma \vdash A \leq B \quad \Gamma \vdash A, B : \star}{\Gamma \vdash M : B} \ \lambda\Pi_{\leq}\text{-conversion}$$

By induction assumption, $FE(\Gamma) \vdash FE_\Gamma(M) : FE_\Gamma(A), FE(\Gamma) \vdash FE_\Gamma(A), FE_\Gamma(B) : \star$. It follows from Lemma 6.3 that $FE(A) =_\beta FE(B)$. Therefore, the judgement $FE(\Gamma) \vdash_{\lambda\Pi} FE(M) : FE(B)$ can be derived by the subsumption rule in $\lambda\Pi$ as follows:

$$\frac{FE(\Gamma) \vdash_{\lambda\Pi} FE(M) : FE(A) \quad FE(A) =_\beta FE(B)}{FE(\Gamma) \vdash_{\lambda\Pi} FE(M) : FE(B)} \ \lambda\Pi\text{-subsumption}$$

□

Lemma 6.5 (Correspondence of reductions between $\lambda\Pi_{\leq}$ and $\lambda\Pi$)

$$\begin{array}{rcl} \Gamma \vdash M : B \ \wedge \ M \rightarrow_\beta M' & \Rightarrow & FE_\Gamma(M) \rightarrow_\beta FE_\Gamma(M') \\ \Gamma \vdash A : K \ \wedge \ A \rightarrow_\beta A' & \Rightarrow & FE_\Gamma(A) \rightarrow_\beta FE_\Gamma(A') \\ \Gamma \vdash K \ \wedge \ K \rightarrow_\beta K' & \Rightarrow & FE_\Gamma(K) \rightarrow_\beta FE_\Gamma(K') \end{array}$$

Proof. Let $M = \mathcal{C}[(\lambda x : A.P)N], M' = \mathcal{C}[P[x := N]]$, then

$$\begin{array}{rcll} FE_\Gamma(M) & = & FE_\Gamma(\mathcal{C})[(\lambda x : FE_\Gamma(A).FE_\Gamma(P))FE_\Gamma(N)] & \\ & \rightarrow_\beta & FE_\Gamma(\mathcal{C})[FE_{\Gamma,x:A}(P)[x := FE_\Gamma(N)]] & \\ & = & FE_\Gamma(\mathcal{C})[FE_\Gamma(P[x := N])] & \textit{Lemma } 6.1 \\ & = & FE_\Gamma(M') & \end{array}$$

The proofs for other two assertions are similar. □

Now the strong normalization follows.

Proposition 6.6 (β-strong normalization) *For β reduction, we have the following results:*

1. *If $\Gamma \vdash K$, then K is strongly normalizing;*
2. *If $\Gamma \vdash A : K$, then A is strongly normalizing;*
3. *If $\Gamma \vdash M : A$, then M is strongly normalizing.*

Proof. By Lemma 6.4, Lemma 6.5, subject reduction and the strong normalization of $\lambda\Pi$. □

7 Justification of Subtyping System

Since our subtyping system is defined over pretypes rather than types, we need to demonstrate that this is a good definition. First, a derivation in a "good" subtyping system should not contain any type which is not well kinded. Second, intuitively valid subtyping rule, including transitivity and reflexivity and some rules specific to the type system, should be admissible.

A possible approach for the justification is to show the equivalence between $\lambda\Pi_{\leq}$ with $\lambda P_{\underline{\leq}}$. We expect the equivalence of subtyping will take the form that, if A, B have the same kind K, then $\Gamma \vdash A \leq B$ hold in our system iff it holds in $\lambda P_{\underline{\leq}}$. But should A, B be kinded in $\lambda P_{\underline{\leq}}$ or in $\lambda\Pi_{\underline{\leq}}$? If the latter choice is assumed, then there the difficulty is to show that the rule in $\lambda\Pi_{\leq}$ is admissible in $\lambda P_{\underline{\leq}}$ since kindings may be different. If kindings are assumed in $\lambda P_{\underline{\leq}}$, then the

subject reduction in $\lambda\Pi_{\leq}$ can not apply and we do not know if kinding can be preserved in the proof where β reduction is involved.

So we take an alternative approach, first we show that our system is equivalent to the system $\lambda P^f_{\leq}$ in which context formation, kinding and typing rules are as in $\lambda\Pi_{\leq}$ but subtyping rules are as in $\lambda P_{\leq}$ except that kindings are fully checked for all subtyping rules. Judgements in $\lambda P^f_{\leq}$ are denoted by $\vdash_f$.

After proving the equivalence between $\lambda\Pi_{\leq}$ and $\lambda P^f_{\leq}$, we can obtain the equivalence between $\lambda\Pi_{\leq}$ and $\lambda P_{\leq}$.

Only subtyping rules of $\lambda P^f_{\leq}$ are listed here.

$$S^f\text{-conv} \quad \frac{\Gamma \vdash_f A, B : K \quad A =_\beta B}{\Gamma \vdash_f A \leq B}$$

$$S^f\text{-trans} \quad \frac{\Gamma \vdash_f A, B, C : K \quad \Gamma \vdash_f A \leq B \quad \Gamma \vdash_f B \leq C}{\Gamma \vdash_f A \leq C}$$

$$S^f\text{-var} \quad \frac{\alpha \ bounded \ in \ \Gamma}{\Gamma \vdash_f \alpha \leq \Gamma(\alpha)}$$

$$S^f\text{-}\pi \quad \frac{\Gamma \vdash_f \pi x{:}A.B, \pi x{:}A'.B' : \star \quad \Gamma \vdash_f A' \leq A, \quad \Gamma, x : A' \vdash_f B \leq B'}{\Gamma \vdash_f \pi x{:}A.B \leq \pi x{:}A'.B'}$$

$$S^f\text{-}\Lambda \quad \frac{\Gamma \vdash_f \Lambda x{:}A.B, \Lambda x{:}A'.B' : K \quad \Gamma, x : A \vdash B \leq B'}{\Gamma \vdash_f \Lambda x{:}A.B \leq \Lambda x{:}A.B'}$$

$$S^f\text{-app} \quad \frac{\Gamma \vdash_f AM, BM : K \quad \Gamma \vdash_f A \leq B}{\Gamma \vdash_f AM \leq BM}$$

Note that $\lambda P^f_{\leq}$ differs from $\lambda P_{\leq}$ also in the subsumption rule where types in subtyping judgement are well kinded.

$$T^f\text{-sub} \quad \frac{\Gamma \vdash_f M : A \quad \Gamma \vdash_f A \leq B \quad \Gamma \vdash_f A, B : \star}{\Gamma \vdash_f M : B}$$

Some kinding checks may be inferred from others in $\lambda P^f_{\leq}$ rules, they are kept here for the purpose of establishing the equivalence with $\lambda\Pi_{\leq}$.

It is easy to verify that the following structural properties (see Section 3) still hold for $\lambda P^f_{\leq}$: generation for kinding, context properties, unicity of kinds, bound β-equivalence, and agreement of judgements. To form such properties in $\lambda P^f_{\leq}$, it suffices to change $\vdash$ to $\vdash_f$ in the original statement. For example, the $\lambda P^f_{\leq}$ version of agreement of judgements becomes:

Proposition 7.1 (Agreement of judgements for $\lambda P^f_{\leq}$)

$$\begin{array}{lcl} \Gamma \vdash_f M : A & \Rightarrow & \Gamma \vdash_f A : \star \\ \Gamma \vdash_f A \leq B & \Rightarrow & \exists K \ \ \Gamma \vdash_f A, B : K \end{array}$$

Proofs for these structural properties in $\lambda P^f_{\leq}$ use straightforward induction on derivations of the concerned judgements. Note that induction are done simultaneously for judgements of formation, kinding, typing and subtyping. While for $\lambda\Pi_{\leq}$, there are two separated inductions, one is simultaneously for the first three judgements, another is for subtyping.

To achieve our goal, the central difficulty is the circularity between kinding, typing and subtyping in the system $\lambda P^f_{\leq}$. We get around the problem by proving the results in the order below.

1. $\Gamma \vdash_f A, B : K \ \wedge \ A =_\beta B \ \Rightarrow \ \Gamma \vdash A \leq B$

2. $\Gamma \vdash_f AM, BM : K \ \wedge \ A \leq B \ \Rightarrow \ \Gamma \vdash AM \leq BM$
3. $\Gamma \vdash_f A \leq B \Rightarrow \Gamma \vdash A \leq B$
4. $\Gamma \vdash A \leq B \ \wedge \ \Gamma \vdash A, B : K \ \Rightarrow \ \Gamma \vdash_f A \leq B$
5. $\Gamma \vdash J \ \Rightarrow \ \Gamma \vdash_f J$
6. $\Gamma \vdash_f J \ \Rightarrow \ \Gamma \vdash J$
7. $\Gamma \vdash_f A \leq B \ \Rightarrow \Gamma \vdash_f A, B : K \ \Rightarrow \ \Gamma \vdash A, B : K$
8. $\Gamma \vdash_f A \leq B \ \Leftrightarrow \ \Gamma \vdash A \leq B \ \wedge \ \Gamma \vdash A, B : K$
9. Equivalence between $\lambda P^f_{\leq}$ and $\lambda P_{\leq}$
10. Equivalence between $\lambda P^f_{\leq}$ and $\lambda\Pi_{\leq}$

where J denotes either a kind K, or a kinding $A : K$, or a typing $M : A$, not including subtyping.

We start by proving that subtyping in the full kinded system $\lambda P^f_{\leq}$ implies subtyping in $\lambda\Pi_{\leq}$. Although $\Gamma \vdash_f A \leq B$ implies $\Gamma \vdash_f A, B : K$, we still do not have the desired property $\Gamma \vdash A, B : K$ at this moment, we need to prove it later by induction on derivation in the context, kinding, typing system.

In order to show that $\lambda P^f_{\leq}$ subtyping implies $\lambda\Pi_{\leq}$ subtyping we need to prove that each subtyping rule of $\lambda P^f_{\leq}$ is admissible in $\lambda\Pi_{\leq}$. It would be desirable to use the results in Section 4. But two of the admissible rules there require the well kindness in $\lambda\Pi_{\leq}$ for types in the subtyping. So these results can not be directly applied to the present situation. However, if these kindings are replaced by kindings in $\lambda P^f_{\leq}$, these rules are still admissible.

The following two propositions are proved with similar methods as proposition 4.2 and proposition 4.7 , using structural properties of $\lambda P^f_{\leq}$.

Proposition 7.2 (Reflexivity with $\lambda P^f_{\leq}$ kinding)

$$\Gamma \vdash_f A, B : K \ \wedge A =_\beta B \ \Rightarrow \ \Gamma \vdash A \leq B$$

Proposition 7.3 (Subtyping family application with $\lambda P^f_{\leq}$ kinding)

$$\Gamma \vdash_f AM, BM : K \ \wedge \ \Gamma \vdash A \leq B \ \Rightarrow \ \Gamma \vdash AM \leq BM$$

It follows that the derivability of subtyping in $\lambda P^f_{\leq}$ implies that of $\lambda\Pi_{\leq}$.

Proposition 7.4 ($\lambda P^f_{\leq}$ subtyping implies $\lambda\Pi_{\leq}$ subtyping)

$$\Gamma \vdash_f A \leq B \quad \Rightarrow \quad \Gamma \vdash A \leq B$$

Proof. By the agreement of judgements for $\lambda P^f_{\leq}$ (Proposition 7.1), $\Gamma \vdash_f A \leq B \Rightarrow \Gamma \vdash_f A, B : K$. The result follows from proposition 7.2,proposition 7.3) and the transitivity (Proposition 4.5). □

The proof in the opposite direction is the key step towards our goal. Note that we can not directly use induction on the subtyping derivation because we do not know if kinding , $\Gamma \vdash A : K$, in $\lambda\Pi_{\leq}$ implies kinding, $\Gamma \vdash_f A : K$, in $\lambda P^f_{\leq}$ yet. Instead, we transform a proof in $\lambda\Pi_{\leq}$ to the one in $\lambda P^f_{\leq}$.

Proposition 7.5 ($\lambda\Pi_{\leq}$ subtyping implies $\lambda P^f_{\leq}$ subtyping)

$$\Gamma \vdash A \leq B \ \wedge \ \Gamma \vdash A, B : K \ \Rightarrow \ \Gamma \vdash_f A \leq B$$

Proof. Consider the subtyping system $\lambda P^+_{\leq}$ obtained from adding all kinding premises to subtyping rules of $\lambda\Pi_{\leq}$.

$$S^+\text{-}\pi \quad \frac{\Gamma \vdash \pi x{:}A.B, \pi x{:}A'.B' : \star \quad \Gamma \vdash^+ A' \leq A \quad \Gamma, x : A' \vdash^+ B \leq B'}{\Gamma \vdash^+ \pi x{:}A.B \leq \pi x{:}A'.B'}$$

$$S^+\text{-}\Lambda \quad \frac{\Gamma \vdash \Lambda x{:}A.B, \Lambda x{:}A'.B' : K \quad A' =_\beta A \quad \Gamma, x : A' \vdash^+ B \leq B'}{\Gamma \vdash^+ \Lambda x{:}A.B \leq \Lambda x{:}A'.B'}$$

$$S^+\text{-ApR} \quad \frac{\Gamma \vdash \alpha M_1..M_n, \alpha M'_1..M'_n : K \quad M_1 =_\beta M'_1 \ \cdots \ M_n =_\beta M'_n}{\Gamma \vdash^+ \alpha M_1..M_n \leq \alpha M'_1..M'_n}$$

$$S^+\text{-ApT} \quad \frac{\Gamma \vdash \alpha M_1..M_n, A : K \quad \Gamma \vdash^+ \Gamma(\alpha) M_1..M_n \leq A}{\Gamma \vdash^+ \alpha M_1..M_n \leq A}$$

$$S^+\text{-ApSL} \quad \frac{\Gamma \vdash (\Lambda x{:}A.B) M_1..M_n, C : K \quad \Gamma \vdash^+ B[x := M_1] M_2..M_n \leq C}{\Gamma \vdash^+ (\Lambda x{:}A.B) M_1..M_n \leq C}$$

$$S^+\text{-ApSR} \quad \frac{\Gamma \vdash C, (\Lambda x{:}A.B) M_1..M_n : K \quad \Gamma \vdash^+ C \leq B[x := M_1] M_2..M_n}{\Gamma \vdash^+ C \leq (\Lambda x{:}A.B) M_1..M_n}$$

It follows from the subject reduction (Proposition 5.2), generation for kinding (Proposition 3.2) and bound β-equivalence (Proposition 3.6) that

$$\Gamma \vdash^+ A \leq B \ \Leftrightarrow \ \Gamma \vdash A, B : K \ \wedge \ \Gamma \vdash A \leq B$$

Each subtyping rule in $\lambda\Pi^+_{\leq}$ can be replaced by a derivation in the following subtyping system.

$$S^{f'}\text{-conv} \quad \frac{\Gamma \vdash A, B : K \quad A =_\beta B}{\Gamma \vdash_{f'} A \leq B}$$

$$S^{f'}\text{-trans} \quad \frac{\Gamma \vdash A, B, C : K \quad \Gamma \vdash_{f'} A \leq B \quad \Gamma \vdash_{f'} B \leq C}{\Gamma \vdash_{f'} A \leq C}$$

$$S^{f'}\text{-var} \quad \frac{\alpha \ \textit{bounded in} \ \Gamma}{\Gamma \vdash_{f'} \alpha \leq \Gamma(\alpha)}$$

$$S^{f'}\text{-}\pi \quad \frac{\Gamma \vdash \pi x{:}A.B, \pi x{:}A'.B' : \star \quad \Gamma \vdash_{f'} A' \leq A, \quad \Gamma, x : A' \vdash_{f'} B \leq B'}{\Gamma \vdash_{f'} \pi x{:}A.B \leq \pi x{:}A'.B'}$$

$$S^{f'}\text{-}\Lambda \quad \frac{\Gamma \vdash \Lambda x{:}A.B, \Lambda x{:}A'.B' : K \quad \Gamma, x : A \vdash_{f'} B \leq B'}{\Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A.B'}$$

$$S^{f'}\text{-app} \quad \frac{\Gamma \vdash AM, BM : K \quad \Gamma \vdash_{f'} A \leq B}{\Gamma \vdash_{f'} AM \leq BM}$$

For example, the rule (S^+-ApSR) could be replaced by

$$\frac{\Gamma \vdash KJ1 \quad \Gamma \vdash_{f'} C \leq B[x := M_1] M_2..M_n \qquad \dfrac{\Gamma \vdash KJ2 \quad B[x := M_1] M_2..M_n =_\beta (\Lambda x{:}A.B) M_1..M_n}{\Gamma \vdash_{f'} B[x := M_1] M_2..M_n \leq (\Lambda x{:}A.B) M_1..M_n} \ S^{f'}\text{-conv}}{\Gamma \vdash_{f'} C \leq (\Lambda x{:}A.B) M_1..M_n} \ trans$$

where $KJ1 \equiv C, B[x := M_1] M_2..M_n, (\Lambda x{:}A.B) M_1..M_n : K, KJ2 \equiv B[x := M_1] M_2..M_n, (\Lambda x{:}A.B) M_1..M_n : K$. The kinding for $B[x := M_1] M_2..M_n$ in the proof is obtained from subject reduction. Therefore, by induction on derivation of $\Gamma \vdash^+ A \leq B$, we have

$$\Gamma \vdash^+ A \leq B \ \Rightarrow \ \Gamma \vdash_{f'} A \leq B$$

Replace all $\vdash, \vdash_{f'}$ by $\vdash_f$ in the proof of $\Gamma \vdash_{f'} A \leq B$ in this system, we get a proof of $\Gamma \vdash_f A \leq B$ in the system $\lambda P^f_{\leq}$.

In conclusion, we have

$$\Gamma \vdash A \leq B \ \wedge\ \Gamma \vdash A, B : K \ \Leftrightarrow\ \Gamma \vdash^{+} A \leq B \ \Rightarrow\ \Gamma \vdash_{f'} A \leq B \ \Rightarrow\ \Gamma \vdash_{f} A \leq B$$

□

Now we can prove the equivalence between $\lambda P^{f}_{\leq}$ and $\lambda\Pi_{\leq}$.

Theorem 7.6 (Equivalence between $\lambda P^{f}_{\leq}$ and $\lambda\Pi_{\leq}$)

$$\begin{array}{lcl}
\Gamma \vdash_{f} K & \Leftrightarrow & \Gamma \vdash K \\
\Gamma \vdash_{f} A : K & \Leftrightarrow & \Gamma \vdash A : K \\
\Gamma \vdash_{f} M : A & \Leftrightarrow & \Gamma \vdash M : A \\
\Gamma \vdash_{f} A \leq B & \Leftrightarrow & \Gamma \vdash A \leq B \ \wedge\ \Gamma \vdash A : K \ \wedge\ \Gamma \vdash B : K
\end{array}$$

Proof. The first three judgements are proved for each direction by induction on the derivation. Observe that the two systems have exactly same set of rules for context formation, kinding and typing. The difference is the existence of kinding premises in the subsumption rule. The result follows from proposition 7.5 and proposition 7.4.

For the equivalence of subtyping, it remains to verify that $\Gamma \vdash_{f} A \leq B : K \ \Rightarrow\ \Gamma \vdash A, B : K$, which follows from the agreement of judgement for $\lambda P^{f}_{\leq}$ (Proposition 7.1) and the first part of the theorem. □

Each rule in $\lambda P^{f}_{\leq}$ is a rule in $\lambda P_{\leq}$ with zero or more additional kinding conditions. Therefore, a $\lambda P^{f}_{\leq}$ derivable judgement is also $\lambda P_{\leq}$-derivable. To show the implication on the reverse direction, we verify that, for each rule of $\lambda P_{\leq}$, its assumptions imply kinding conditions in the corresponding $\lambda P^{f}_{\leq}$ rule, using the agreement of judgements for $\lambda P^{f}_{\leq}$ (Proposition 7.1).

We write $\vdash_{AC}$ denote the judgement in Aspinall and Compagnoni's system $\lambda P_{\leq}$.

Proposition 7.7 (Equivalence between $\lambda P^{f}_{\leq}$ and $\lambda P_{\leq}$)

$$\begin{array}{lcl}
\Gamma \vdash_{f} K & \Leftrightarrow & \Gamma \vdash_{AC} K \\
\Gamma \vdash_{f} A : K & \Leftrightarrow & \Gamma \vdash_{AC} A : K \\
\Gamma \vdash_{f} M : A & \Leftrightarrow & \Gamma \vdash_{AC} M : A \\
\Gamma \vdash_{f} A \leq B & \Leftrightarrow & \Gamma \vdash_{AC} A \leq B
\end{array}$$

Proof.
($\Rightarrow$). Simultaneously by induction on derivation.

($\Leftarrow$). Simultaneously by induction on derivation.
Case. (T-sub). Suppose that the derivation ends with

$$\frac{\Gamma \vdash_{AC} M : A \quad \Gamma \vdash_{AC} A \leq B}{\Gamma \vdash_{AC} M : B}\ \text{T-sub}$$

We infer as follows:

$$\begin{array}{lcll}
\Gamma \vdash_{AC} M : A & \Rightarrow & \Gamma \vdash_{f} M : A & IH \\
 & \Rightarrow & \Gamma \vdash_{f} A : \star & prop.\ 7.1 \\
\Gamma \vdash_{AC} A \leq B & \Rightarrow & \Gamma \vdash_{f} A \leq B & IH \\
 & \Rightarrow & \Gamma \vdash_{f} A, B : \star & prop.\ 7.1
\end{array}$$

The result follows by application of T^{f}-sub.

Other cases are similar. □

The equivalence between $\lambda\Pi_{\leq}$ and $\lambda P_{\leq}$ follows from that of $\lambda P^{f}_{\leq}$ and $\lambda P_{\leq}$ and that of $\lambda P^{f}_{\leq}$ and $\lambda\Pi_{\leq}$.

Proposition 7.8 (Equivalence between $\lambda P_{\leq}$ and $\lambda\Pi_{\leq}$)

$$\begin{array}{lcl}
\Gamma \vdash_{AC} K & \Leftrightarrow & \Gamma \vdash K \\
\Gamma \vdash_{AC} A : K & \Leftrightarrow & \Gamma \vdash A : K \\
\Gamma \vdash_{AC} M : A & \Leftrightarrow & \Gamma \vdash M : A \\
\Gamma \vdash_{AC} A \leq B & \Leftrightarrow & \Gamma \vdash A \leq B \ \wedge \ \Gamma \vdash A : K \ \wedge \ \Gamma \vdash B : K
\end{array}$$

Proof. By theorem 7.6 and proposition 7.7. □

8 Decidability and Minimal Typing

The decidability of subtyping can follow from the equivalence between $\lambda\Pi_{\leq}$ and $\lambda P_{\leq}$, but it can also be directly obtained from the subtyping rules of $\lambda\Pi_{\leq}$.

A subtyping rule

$$\frac{A_1 \quad A_2 \quad \cdots \quad A_n}{A}$$

can be turned to a rewriting rule as $A \ \rightarrow \ A_1, \cdots, A_n$.

Thus we can obtain an ordered set of rewriting rules (corresponding the order by which the subtyping rules are written) on sets of subtyping judgements. The resulting subtyping algorithm (i.e., the recursive procedure formed from the ordered rewritting rules) is:

check($\Gamma \vdash S \leq T$) =
  if $S \equiv \pi x : A.B$ *and* $T \equiv \pi x : A'.B'$
    then *check*($\Gamma \vdash A' \leq A$)
      and check($\Gamma, x : A' \vdash B \leq B'$)
  else if $S \equiv \Lambda x : A.B$ *and* $T \equiv \Lambda x : A'.B'$ *and* $A =_{\beta} A'$
    then check($\Gamma, x : A' \vdash B \leq B'$)
  else if $S \equiv \alpha M_1..M_n$ *and* $T \equiv \alpha M'_1..M'_n$ *and* $M_1 =_{\beta} M'_1 \cdots M_n =_{\beta} M'_n$
    then True
  else if $S \equiv (\Lambda x{:}A.B)M_1..M_n$
    then check($\Gamma \vdash B[x := M_1]M_2..M_n \leq T$)
  else if $T \equiv (\Lambda x{:}A.B)M_1..M_n$
    then check($\Gamma \vdash S \leq B[x := M_1]M_2..M_n$)
  else false.

To show the correctness and the termination of the algorithm, we need a measure which is based on the reduction $\rightarrow_{\beta_2\Gamma}$ defined as $\rightarrow_{\beta_2} \cup \rightarrow_{\Gamma}$, where Γ-reduction is the expansion of bounded type variable by their bound:

$$\mathcal{C}[\alpha] \rightarrow_{\Gamma} \mathcal{C}[\Gamma(\alpha)]$$

We write $\rightarrow_R^n$ to indicate that a reduction is n steps long (or $\rightarrow_R^{>n}$ for more than n steps). The $\beta_2\Gamma$-reduction has subject reduction and strong normalization properties:

Lemma 8.1 (Subject reduction for $\rightarrow_{\beta_2\Gamma}$) $\Gamma \vdash A : K \ \wedge \ A \rightarrow_{\beta_2\Gamma} A' \ \Rightarrow \ \Gamma \vdash A' : K$.

Lemma 8.2 (Strong normalization for $\rightarrow_{\beta_2\Gamma}$) *If $\Gamma \vdash A : K$, then there is no infinite $\beta_2\Gamma$-reduction from A.*

Proof. Using the function FE defined in Section 3. Observe that

$$\begin{array}{lll} \Gamma \vdash A : K \ \wedge \ A \rightarrow_\Gamma A' & \Rightarrow & \Gamma \vdash A' : K \ \wedge \ FE(A) = FE(A') \\ \Gamma \vdash A : K \ \wedge \ A \rightarrow_{\beta_2} A' & \Rightarrow & \Gamma \vdash A' : K \ \wedge \ FE(A) \rightarrow_{\beta_2} FE(A') \end{array}$$

We show that a reduction from a well-kinded type A can not have infinitely many β_2-reduction steps. With well-formed context Γ, it is impossible to have an infinite long Γ-reduction either. So any $\beta_2\Gamma$-reduction must be terminate.

Suppose that there were an infinite reduction $\beta_2\Gamma$ started from A:

$$A_0 \rightarrow_{\beta_2\Gamma} A_1 \rightarrow_{\beta_2\Gamma} \cdots \rightarrow_{\beta_2\Gamma} A_n \cdots$$

where $A_0 = A$. Then, by subject reduction property, we have $\Gamma \vdash A, A_1, A_2, .., A_n, : K$. Therefore,

$$FE(A_0), FE(A_1), FE(A_2), .., FE(A_n), ..$$

are well-defined. It follows from Lemma 6.4 that

$$FE(\Gamma) \vdash_{\lambda\Pi} FE(A), FE(A_1), FE(A_2), .., FE(A_n), .. : FE(K)$$

. By the correspondence of reductions between $\lambda\Pi_\leq$ and $\lambda\Pi$ (Lemma 6.5), we have

$$A_i \rightarrow_{\beta_2\Gamma} A_{i+1} \ \Rightarrow \ FE(A_i) \rightarrow_{\beta_2} FE(A_{i+1}) \ \vee \ FE(A_i) = FE(A_{i+1})$$

Let $\rightarrow_{\beta_2 id}$ denote the relation $\beta_2 \cup id$ where id is the identity relation. Then we have a reduction:

$$FE(A_0) \rightarrow_{\beta_2 id} FE(A_1) \rightarrow_{\beta_2 id} FE(A_2) \rightarrow_{\beta_2 id} \cdots FE(A_n) \rightarrow_{\beta_2 id} \cdots$$

Thus, we can extract from it a subsequence of β_2 reduction:

$$FE(A_{i_0}) \rightarrow_{\beta_2} FE(A_{i_1}) \rightarrow_{\beta_2} FE(A_{i_2}) \rightarrow_{\beta_2} \cdots FE(A_{i_n}) \rightarrow_{\beta_2} \cdots$$

such that $A_0 = A_{i_0}$ and $FE(A_j) = FE(A_{i_k})$ for $i_k \leq j < i_{k+1}$. Since $\lambda\Pi$ is strongly normalizaing, this reduction sequence can not be infinitely long. Hence, there is a m such that there are no β_2 reduction from $FE(A_m)$. It follows that there are no β_2 reduction from A_m in $\lambda\Pi_\leq$.

If the original reduction from A is infinite, then from A_m there is an infinite Γ reduction. Since A_m is well-kinded, this is impossible.

□

Remark 8.3 *The $\beta_2\Gamma$-reduction is a complicated process as β_2-reduction may introduce new Γ redex and Γ-reduction may introduce new β_2 redex. To illustrate such difficulty and help understanding the above proof, we give an example as follows:*

Assume a context $\Gamma = A : \star, \alpha \leq \alpha', \alpha' \leq \Lambda x{:}A.A, y : A, a : \alpha y, u : \alpha y \rightarrow \alpha y, D : \alpha \rightarrow \star$. *Consider a well-kinded term*

$$(\Lambda u{:}\alpha y.D(u(u(a))))(\lambda z{:}\alpha y.z)$$

and a $\beta_2\Gamma$-reduction from this term:

$$\begin{array}{lll} & \Lambda u{:}\alpha y.D(u(u(a))))(\lambda z{:}\alpha y.z) & \\ \rightarrow_{\beta_2} & D((\lambda z{:}\alpha y.z)((\lambda z{:}\alpha y.z)(a))) & \Gamma \textit{ redex are increased} \\ \rightarrow_{\Gamma} & D((\lambda z{:}\alpha' y.z)((\lambda z{:}\alpha y.z)(a))) & \\ \rightarrow_{\Gamma} & D((\lambda z{:}(\Lambda x{:}A.A)y.z)((\lambda z{:}\alpha y.z)(a))) & a\ \beta_2 \textit{ redex is created} \\ \rightarrow_{\beta_2} & D((\lambda z{:}A.z)((\lambda z{:}\alpha y.z)(a))) & \\ \cdots & & \end{array}$$

By the function FE, the above reduction is transformed to a reduction in $\lambda\Pi$ as follows:

$$\begin{array}{lll}
 & FE(\Lambda u{:}\alpha y.D(u(u(a)))(\lambda z{:}\alpha y.z)) & = (\Lambda u{:}(\Lambda x{:}A.A)y.D(u(u(a)))(\lambda z{:}(\Lambda x{:}A.A)y.z) \\
\rightarrow_{\beta_2} & FE(D((\lambda z{:}\alpha y.z)((\lambda z{:}\alpha y.z)(a)))) & = D((\lambda z{:}(\Lambda x{:}A.A)y.z)((\lambda z{:}(\Lambda x{:}A.A)y.z)(a))) \\
\equiv & FE(D((\lambda z{:}\alpha' y.z)((\lambda z{:}\alpha y.z)(a)))) & = D((\lambda z{:}(\Lambda x{:}A.A)y.z)((\lambda z{:}(\Lambda x{:}A.A)y.z)(a))) \\
\equiv & FE(D((\lambda z{:}(\Lambda x{:}A.A)y.z)((\lambda z{:}\alpha y.z)(a)))) & = D((\lambda z{:}(\Lambda x{:}A.A)y.z)((\lambda z{:}(\Lambda x{:}A.A)y.z)(a))) \\
\rightarrow_{\beta_2} & FE(D((\lambda z{:}A.z)((\lambda z{:}\alpha y.z)(a)))) & = D((\lambda z{:}A.z)((\lambda z{:}(\Lambda x{:}A.A)y.z)(a))) \\
\cdots & &
\end{array}$$

This shows that the $\beta_2\Gamma$-reduction in $\lambda\Pi_{\leq}$ has been transformed to a β_2-reduction in $\lambda\Pi$. Hence the termination of the former can be derived from that of the later.

Now we show the correctness of this algorithm.

Proposition 8.4 (Correctness of the subtyping algorithm)

$$\Gamma \vdash S \leq T \;\Leftrightarrow\; check(\Gamma \vdash S \leq T) = True$$

Proof.

(<=). Immediate.

(=>). Induction on the derivation of $\Gamma \vdash S \leq T$. Analyse according to the last rule in the derivation.

Case S-π. Let the subtyping judgement be $\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'$. Then the algorithm can only use S-π as the rewritting rule. That is, $check(\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B')$ succeeds iff both $check(\Gamma \vdash A' \leq A)$ and $check(\Gamma, x : A' \vdash B \leq B')$ succeed. The rewritting will succeed by induction assumption.

Case S-Λ. Similar.

Case S-ApR. Similar.

Case S-ApT. If the subtyping judgement is of the form $\Gamma \vdash \alpha M_1..M_n \leq \alpha M'_1..M'_n$ and $M_1 =_\beta M'_1 \cdots M_n =_\beta M'_n$, then the algorithm will succeed by using the rule S-ApR. Otherwise, the rule S-ApT will be used for rewritting. By induction assumption, the algorithm will succeed.

Case S-ApSL. The algorithm will only select the rule S-ApSL for rewritting, so it will succeed by induction assumption.

Case S-ApSR. This is the most complicated case. The subtyping judgement may be rewritted by S-ApT, S-ApSL and S-ApSR. Define a relation $\rightarrow_R$ on subtyping judgments:

$$\begin{array}{rcl}
\Gamma \vdash \alpha M_1..M_n \leq A & \rightarrow_R & \Gamma \vdash \Gamma(\alpha)M_1..M_n \leq A \\
\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq C & \rightarrow_R & \Gamma \vdash B[x := M_1]M_2..M_n \leq C \\
\Gamma \vdash C \leq (\Lambda x{:}A.B)M_1..M_n & \rightarrow_R & \Gamma \vdash C \leq B[x := M_1]M_2..M_n
\end{array}$$

satisfies the diamond property:

$$\begin{array}{ll}
 & \Gamma \vdash A_1 \leq B_1 \rightarrow_R \Gamma \vdash A_2 \leq B_2 \;\wedge\; \Gamma \vdash A_1 \leq B_1 \rightarrow_R \Gamma \vdash A_3 \leq B_3 \\
\Rightarrow & \exists A_4, B_4 \; s.t. \; \Gamma \vdash A_2 \leq B_2 \rightarrow_R \Gamma \vdash A_4 \leq B_4 \;\wedge\; \Gamma \vdash A_3 \leq B_3 \rightarrow_R \Gamma \vdash A_4 \leq B_4
\end{array}$$

Therefore, $\rightarrow_R$ is confluent. By the strong normalization of $\beta_2\Gamma$-reduction, $\rightarrow_R$ is also a terminate relation. The result follows. □

By the subject reduction and strong normalization of $\beta_2\Gamma$-reduction, it makes sense to define a function $MaxRed(A)$ which is the maximal number of $\rightarrow_{\beta_2\Gamma}$ reductions from A:

$$MaxRed(A) \;=_{def}\; max\{\; n \;|A \rightarrow^n_{\beta_2\Gamma} A^{\beta_2\Gamma}\}$$

Recall that A^R indicates the R normal form of A and $\rightarrow^n_R$ is n step R-reductions.

The function $MaxRed$ has desired properties:

Lemma 8.5

1. $MaxRed(\Gamma(\alpha)M_1..M_n) < MaxRed(\alpha M_1..M_n)$;
2. $MaxRed((\Lambda x : A.B)M_1..M_n) < MaxRed(B[x := M_1]M_2..M_n)$.

Then define a measure $weight(A, B)$ of two types A, B as the pair: $weight(A, B) =_{def} < MaxRed(A) + MaxRed(B), Size(A) + Size(B) >$. The termination of the algorithm follows.

Proposition 8.6 (Decidability of subtyping) *If* $\Gamma \vdash A : K_1, \Gamma \vdash B : K_2$, *then the subtyping judgement* $\Gamma \vdash A \leq B$ *is decidable.*

Proof. By inspecting the rules. $check(\Gamma \vdash A \leq B)$ will terminate by the measure $weight(A, B)$. □

The next step towards proving decidability is to design algorithmic versions of the remaining judgements (typing, kinding and context formation). Appendix B shows the algorithmic rules, below we just give highlights. Judgements in algorithmic rules are denoted by $\Gamma \vdash_{\mathcal{A}} J$, with the convention that premises are evaluated in order, the rules form an algorithm.

The main idea of forming the algorithmic rules is to remove the subsumption rule and to modify the application rule so that it takes the subtyping into account. Thus if we know the terms M, N are typed as $\Gamma \vdash M : A, \Gamma \vdash N : B$, we need to infer a type C such that $\Gamma \vdash MN : C$. When MN is well typed, the type A of M should be equivalent to a type of the form $\pi x{:}D.E$, but the actual form of A may be $\alpha M_1..M_n$ or $(\Lambda y{:}D.E)M_1..M_n$. Therefore, we need to infer from the type of M a type of the form $\pi x{:}D.E$. In $\lambda P_{\leq}$, this is achieved by using a function FLUB which returns the π-type from the input type. We take a different approach by introducing a new judgement

$$\Gamma \vdash A \leq_{\pi lub} B$$

to express the fact that B is the least π-type of type A. The rules for this new judgement is simply a subset of subtyping rules plus a reflexive rule

$$\text{Lub-Refl} \quad \frac{}{\Gamma \vdash \pi x{:}A.B \leq_{\pi lub} \pi x{:}A.B}$$

$$\text{Lub-ApT} \quad \frac{\Gamma \vdash \Gamma(\alpha)M_1..M_n \leq_{\pi lub} A}{\Gamma \vdash \alpha M_1..M_n \leq_{\pi lub} A}$$

$$\text{Lub-ApSL} \quad \frac{\Gamma \vdash B[x := M_1]M_2..M_n \leq_{\pi lub} C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq_{\pi lub} C}$$

For a type A convertible to a π-type, the unique minimal π-type upper bound of A can be derived by above rules.

Proposition 8.7 (Properties of πlub judgement)

1. $\Gamma \vdash A \leq_{\pi lub} B \Rightarrow \Gamma \vdash A \leq B$
2. $\Gamma \vdash A \leq \pi x{:}B.C \Rightarrow \exists B', C'$ *s.t.* $\Gamma \vdash A \leq_{\pi lub} \pi x{:}B'.C' \wedge \Gamma \vdash \pi x{:}B'.C' \leq \pi x{:}B.C$
3. *Given a type* A, *it is decidable if there is a type* $\pi x{:}B.C$ *such that* $\Gamma \vdash A \leq_{\pi lub} \pi x{:}B.C$ *is derivable.*
4. $\Gamma \vdash A \leq_{\pi lub} B \wedge \Gamma \vdash A : K \Rightarrow \Gamma \vdash B : K$
5. $\Gamma \vdash A \leq \pi x{:}B.C \wedge \Gamma \vdash A \leq \pi x{:}B'.C' \Rightarrow B = B' \wedge C = C'$

Proof. The last claim is proved by induction on the derivation of $\Gamma \vdash A \leq_{\pi lub} B$ using subject reduction. The others are straightforward. □

Another modification from original rule to the algorithmic one, as did in $\lambda P_{\leq}$, is the removal of formation from (T-var) and (K-var) so that typing and kinding become independent of context formation. As a consequence, $\Gamma \vdash_{\mathcal{A}} J$ may not imply $\Gamma \vdash \star$. So additional kinding checks $\Gamma \vdash_{\mathcal{A}} A : \star$ should be added to abstraction rules.

The difference between our approach and $\lambda P_{\leq}$ is in the typing and kinding application rules. We use subtyping directly without β_2 normalization. The FLUB function in $\lambda P_{\leq}$ has been replaced by the πlub judgement.

Unlike in the original system, there are no kinding premises in K-app, T-app for types participating subtyping because their well kindness can be inferred.

The algorithmic rules for kinding and typing applications are:

$$\text{AK-app} \qquad \frac{\Gamma \vdash_{\mathcal{A}} A : \Pi x{:}B.K \quad \Gamma \vdash_{\mathcal{A}} M : B' \quad \Gamma \vdash B' \leq B}{\Gamma \vdash_{\mathcal{A}} AM : K[x := M]}$$

$$\text{AT-app} \qquad \frac{\Gamma \vdash_{\mathcal{A}} M : A \quad \Gamma \vdash A \leq_{\pi lub} \pi x : B.C \quad \Gamma \vdash_{\mathcal{A}} N : B' \quad \Gamma \vdash B' \leq B}{\Gamma \vdash_{\mathcal{A}} MN : C[x := N]}$$

Proofs for soundness and completeness of algorithmic system are little longer than those in $\lambda P_{\leq}$. But there are no specific tricks involved.

For the proof of the soundness of algorithmic rules, we need to ensure that the well kindness of types in premises of subsumption can be satisfied. This is achieved by using the property of πlub judgement and generation for kinding.

Proposition 8.8 (Soundness of algorithmic system) *For all* Γ, A, K, M,

$$\begin{array}{lll} 1. & \Gamma \vdash_{\mathcal{A}} K & \Rightarrow \quad \Gamma \vdash K \\ 2. & \Gamma \vdash \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} A : K & \Rightarrow \quad \Gamma \vdash A : K \\ 3. & \Gamma \vdash \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} M : A & \Rightarrow \quad \Gamma \vdash M : A \end{array}$$

Proof. Simultaneously by induction on the derivation in the algorithmic system.

Case (AT-app). Suppose we have a derivation ended by an application of the rule AT-app

$$\frac{\Gamma \vdash_{\mathcal{A}} M : A \quad \Gamma \vdash A \leq_{\pi lub} \pi x : B.C \quad \Gamma \vdash_{\mathcal{A}} N : B' \quad \Gamma \vdash B' \leq B}{\Gamma \vdash_{\mathcal{A}} MN : C[x := N]} \text{ AT-app}$$

We infer as follows

$$\begin{array}{lcll} \Gamma \vdash \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} M : A & \Rightarrow & \Gamma \vdash M : A & \textit{induction assumption} \\ & \Rightarrow & \Gamma \vdash A : \star & \textit{prop.}\ 3.10 \\ \Gamma \vdash A \leq_{\pi lub} \pi x{:}B.C & \Rightarrow & \Gamma \vdash A \leq \pi x{:}B.C & \textit{prop.}\ 8.7(1) \\ \Gamma \vdash \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} N : B' & \Rightarrow & \Gamma \vdash N : B' & IH \\ & \Rightarrow & \Gamma \vdash B' : \star & \textit{prop.}\ 3.10 \\ \Gamma \vdash A \leq_{\pi lub} \pi x{:}B.C \ \wedge \ \Gamma \vdash A : \star & \Rightarrow & \Gamma \vdash \pi x{:}B.C : \star & \textit{prop.}\ 8.7(4) \\ & \Rightarrow & \Gamma \vdash B : \star & \textit{prop.}\ 3.2,\ 3.3 \end{array}$$

So we have a derivation ending by

$$\dfrac{\dfrac{\Gamma \vdash M : A \quad \Gamma \vdash A \leq \pi x{:}B.C \quad \Gamma \vdash A, \pi x{:}B.C : \star}{\Gamma \vdash M : \pi x{:}B.C} \text{ T-sub} \quad \dfrac{\Gamma \vdash N : B' \quad \Gamma \vdash B' \leq B \quad \Gamma \vdash B', B : \star}{\Gamma \vdash N : B} \text{ T-sub}}{\Gamma \vdash MN : C[x := N]} \text{ T-app}$$

The case for (AK-app) is similar. Others are easy. □

Corollary 8.9

$$\Gamma \vdash_{\mathcal{A}} \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} A : K \ \Rightarrow \ \Gamma \vdash A : K$$
$$\Gamma \vdash_{\mathcal{A}} \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} M : A \ \Rightarrow \ \Gamma \vdash M : A$$

Due to the lack of kinding conversion and typing subsumption in the algorithmic rules, generally we do not have

$$\begin{array}{lcl} \Gamma \vdash A : K & \Rightarrow & \Gamma \vdash_{\mathcal{A}} A : K \\ \Gamma \vdash M : A & \Rightarrow & \Gamma \vdash_{\mathcal{A}} M : A \end{array}$$

Instead we have the completeness result in the following sense.

Proposition 8.10 (Completeness of algorithmic system)

$$\begin{array}{lcl} 1. \ \Gamma \vdash K & \Rightarrow & \Gamma \vdash_{\mathcal{A}} K \\ 2. \ \Gamma \vdash A : K & \Rightarrow & \exists K_a \ s.t. \ \Gamma \vdash_{\mathcal{A}} A : K_a \ \wedge \ K_a =_\beta K \ \wedge \ \Gamma \vdash K_a \\ 3. \ \Gamma \vdash M : A & \Rightarrow & \exists A_a \ s.t. \ \Gamma \vdash_{\mathcal{A}} M : A_a \ \wedge \ \Gamma \vdash A_a \leq A \end{array}$$

Proof. Simultaneously by induction on derivations in the original system.
Case (K-app). Suppose the last step of derivation is

$$\frac{\Gamma \vdash A : \Pi x{:}B.K \quad \Gamma \vdash M : B}{\Gamma \vdash AM : K[x := M]} \ K-app$$

We infer as follows

$\Gamma \vdash A : \Pi x{:}B.K$	$\Rightarrow$	$\exists K_a \ s.t. \ \Gamma \vdash_{\mathcal{A}} A : K_a \ \wedge \ K_a =_\beta \Pi x{:}B.K$	IH
	$\Rightarrow$	$\exists B', K' \ s.t. \ K_a = \Pi x{:}B'.K' \wedge$ $B' =_\beta B \ \wedge \ K' =_\beta K$	$confluence$
$\Gamma \vdash M : B$	$\Rightarrow$	$\Gamma \vdash_{\mathcal{A}} M : B'' \ \wedge \ \Gamma \vdash B'' \leq B$	IH
$\Gamma \vdash M : B$	$\Rightarrow$	$\Gamma \vdash \star \ \wedge \ \Gamma \vdash B : \star$	$prop$ 3.3, $prop.$ 3.10
$\Gamma \vdash \star \ \wedge \ \Gamma \vdash_{\mathcal{A}} A : K_a$	$\Rightarrow$	$\Gamma \vdash A : K_a$	$prop$ 8.8
	$\Rightarrow$	$\Gamma \vdash K_a$	$prop.$ 3.10
	$\Rightarrow$	$\Gamma \vdash B' : \star$	F-Π, $prop.$ 3.10
$\Gamma \vdash B, B' : \star \ \wedge \ B' =_\beta B$	$\Rightarrow$	$\Gamma \vdash B \leq B'$	$prop.$ 4.2
	$\Rightarrow$	$\Gamma \vdash B'' \leq B'$	$prop.$ 4.5

So we can have a derivation ending by

$$\frac{\Gamma \vdash_{\mathcal{A}} A : \Pi x{:}B'.K' \quad \Gamma \vdash_{\mathcal{A}} M : B'' \quad \Gamma \vdash B'' \leq B'}{\Gamma \vdash_{\mathcal{A}} AM : K'[x := M]} \ \text{AK-app}$$

and $K'[x := M] =_\beta K[x := M]$.

Case. (T-app). Suppose the derivation is ended with

$$\frac{\Gamma \vdash M : \pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]} \ \text{T-app}$$

Then there C, A', B', A'' such that

$\Gamma \vdash M : \pi x{:}A.B$	$\Rightarrow$	$\Gamma \vdash_{\mathcal{A}} M : C \ \wedge \ \Gamma \vdash C \leq \pi x{:}A.B$	IH
$\Gamma \vdash C \leq \pi x{:}A.B$	$\Rightarrow$	$\Gamma \vdash C \leq_{\pi lub} \pi x{:}A'.B' \ \wedge \ \Gamma \vdash \pi x{:}A'.B' \leq \pi x{:}A.B$	$prop$ 8.7
	$\Rightarrow$	$\Gamma \vdash A \leq A'$	$prop$ 3.2
$\Gamma \vdash N : A$	$\Rightarrow$	$\Gamma \vdash_{\mathcal{A}} N : A'' \ \wedge \ \Gamma \vdash A'' \leq A$	IH
	$\Rightarrow$	$\Gamma \vdash A'' \leq A'$	$prop$ 4.5

Therefore, we can have a derivation ending by

$$\frac{\Gamma \vdash_{\mathcal{A}} M : C \quad \Gamma \vdash C \leq_{\pi lub} \pi x{:}A'.B' \quad \Gamma \vdash_{\mathcal{A}} N : A'' \quad \Gamma \vdash A'' \leq A'}{\Gamma \vdash_{\mathcal{A}} MN : B'[x := N]} \text{ AT-app}$$

and $\Gamma \vdash B'[x := N] \leq B[x := N]$ follows from the substitution (Proposition 3.9).

Other cases are easy. □

In the proof of completeness, we have used reflexivity and transitivity of subtyping. To apply reflexivity, we need kinding condition, which is proved by context properties, agreement of judgments and soundness of the algorithmic rules. Generation for kinding has been used to break the subtyping between π-types so that the transitivity can apply.

A minimal type of a term M under context Γ is a type A such that $\Gamma \vdash M : A$ and for any type B, the judgement $\Gamma \vdash M : B$ implies $\Gamma \vdash B \leq A$. Note that, by this definition, the minimal type of a term M may not be unique: B is a minimal type of M iff any well-formed type which is β-equivalent to B is also a minimal type of M. But our algorithmic system will always return the same minimal type. This type may not be in normal form, for example, if $x : A \in \Gamma$ and A, then the algorithm will return A for x, but A may not be in normal form. To obtain the minimal normalized type of a term, one can simply normalize the type returned by the algorithm.

In order to show the minimal typing property, it remains to show that the A_a in the proposition of completeness is independent of the specific type A. To this aim, it suffices to prove the uniqueness of algorithmic typing.

Proposition 8.11 (Uniqueness and minimality of algorithmic typing)

$$\Gamma \vdash_{\mathcal{A}} M : A \ \wedge \ \Gamma \vdash_{\mathcal{A}} M : B \ \Rightarrow \ A = B$$
$$\Gamma \vdash_{\mathcal{A}} M : A \ \wedge \ \Gamma \vdash M : B \ \Rightarrow \ \Gamma \vdash A \leq B$$

Proof. Part 1 is proved by induction on the size of M, using uniqueness of πlub (Proposition 8.7). Part 2 follows from part 1. □

The minimal typing property follows.

Corollary 8.12 (Minimal typing property for $\lambda\Pi_{\leq}$)

$$\Gamma \vdash M : A \ \Rightarrow \ (\exists B \ s.t. \ \Gamma \vdash M : B \ \wedge \ \forall C \ \ \Gamma \vdash M : C \Rightarrow \Gamma \vdash B \leq C)$$

Decidability is summarized as follows.

Proposition 8.13 (Decidability of algorithmic $\lambda\Pi_{\leq}$) *The following problems are decidable: for all Γ, K, M, A and B,*

1. $\Gamma \vdash A : K \ \wedge \ \Gamma \vdash B : K' \ \Rightarrow \ \Gamma \vdash A \leq B$ *?*
2. $\Gamma \vdash \star \ \Rightarrow \ \exists K_a \ s.t. \ \Gamma \vdash_{\mathcal{A}} A : K_a$ *?*
3. $\Gamma \vdash \star \ \Rightarrow \ \exists A_a \ s.t. \ \Gamma \vdash_{\mathcal{A}} M : A_a$ *?*
4. $\Gamma \vdash_{\mathcal{A}} K$ *?*

Proof. The assertions can be proved in the order shown. Part 1 has been proved in Proposition 8.6. □

By the equivalence between algorithmic rules and the original one, we obtain the decidability of judgements in $\lambda\Pi_{\leq}$.

Proposition 8.14 (Decidability of $\lambda\Pi_{\leq}$) *The following problems are decidable: for all Γ, K, M, A and B,*

1. $\Gamma \vdash A : K \ \wedge \ \Gamma \vdash B : K' \ \Rightarrow \ \Gamma \vdash A \leq B$?
2. $\Gamma \vdash \star \ \Rightarrow \ \exists K_a \ s.t. \ \Gamma \vdash A : K_a$?
3. $\Gamma \vdash \star \ \Rightarrow \ \exists A_a \ s.t. \ \Gamma \vdash M : A_a$?
4. $\Gamma \vdash K$?

The algorithm for checking subtyping, typing, kinding and context formation can be obtained by orienting the algorithmic rules and subtypings rules into ordered rewriting rules. But the obtained algorithm still have some inefficiency. For example, in order to check the subtyping of two equal types, say $(\Lambda x{:}A.B)M \leq (\Lambda x{:}A.B)M$, the algorithm will actually β_2-normalize these two types in the checking process. But we can add an equality reflexivity rule, which is assumed of highest priority among all subtyping rules:

$$\frac{}{\Gamma \vdash A \leq A} \ EqRefl$$

Thus unecessary β_2-normalizations can be eliminated in the type checking.

9 Concluding Remarks and Related Works

The study of subtyping extension of dependent types dates back to the early work of Cardelli [Car87], who gave basic definitions and ideas about semi-decision procedures. Since then, adding restricted subtyping to dependent types with various features have been investigated by several authors, e.g. [Coq92, Luo96, Pfe89]. But the extension of subtyping in its general form to first order dependent types has been achieved only recently by Aspinall and Compagnoni[AC96]. From methodological point of view, the major technical invention of [AC96] is the algorithmic subtyping system, which has transitivity elimination property and thus can be considered as the key step towards the proofs of subject reduction and decidability of subtyping. Transitivity elimination is achieved mainly by the introduction of two rules S-ApR and S-ApT. The former is adopted to recover the conversion at β_2 normalized type level. The later resolves the conflit between transitivity elimination and application of subtyping declaration.

Our work goes a step further in building a system with transitivity elimination at type level by introducing a pair of rules S-ApSR and S-ApSL.

Transitivity elimination has been one of the main research subjects in the history of subtyping. Among subtyping systems in λ-cube[Bar92], apart from the simply typed λ-calculus, the work on $F_{\leq}$ by Curien and Ghelli[CG92] is the first one having this property. In that work, they have shown how to overcome the conflit between transitivity elimination and the rule

$$\frac{\alpha \leq A \ \in \Gamma}{\Gamma \vdash \alpha \leq A}$$

Mitchell [Mit88] has proposed a subtyping system for second order λ-calculus, where the transitivity elimination can not be obtained because there are some rules like

$$\frac{}{\forall X.A \leq B[A/X]}$$

In [LMS95], Longo, Milsted and Soloviev have proposed a system $Co^{\vdash}$, which is equivalent to that of Mitchell, but it has transitivity elimination property.

In the studies of subtyping extensions of F^{ω} and λP, there is a new obstacle to transitivity elimination, that is, the type conversion. The existing works $F^{\omega}_{\leq}$[SP94],$F^{\omega}_{\wedge}$[Com94] and $\lambda P_{\leq}$ have

developed techniques to achieve transitivity elimination at the level of normalized types. Our system is the first one having type level transitivity elimination among subtyping systems with type conversions. This technique is applicable to other systems with type conversions.

One feature of this work is the independent pretype based subtyping relation, the subtyping system is separated from typing, kinding and formation. As a consequent, meta-theoretic properties, like subject reduction, could be proved with less difficulty than in $\lambda P_{\leq}$.

It should be pointed out that the pretype based subtyping can not be formed simply by dropping kinding premises from subtyping rules in $\lambda P_{\leq}$, since transitivity can not be eliminated. If $\lambda P_{\leq}$ did not have kinding checks in subtyping rules, then we could not guarantee the kindability of the type B in the following transitivity application

$$\frac{\Gamma \vdash A \leq B \quad \Gamma \vdash B \leq C}{\Gamma \vdash A \leq C}\ trans$$

even if types A, C in the last judgment were well-kinded. That is, a subtyping derivation with a well-formed judgement at the end would have an ill-kinded type in the middle. Such derivation would not be considered as justified.

Subtyping in system $F^{\omega}_{\leq}$ developed by Martin Steffen and Benjamin Pierce [SP94] is also based on pretypes, but there are still kinding premises in the the subtyping rule. For example, the transitivity rule in $F^{\omega}_{\leq}$ is:

$$\frac{\Gamma \vdash B : K \quad \Gamma \vdash A \leq B \quad \Gamma \vdash B \leq C}{\Gamma \vdash A \leq C}\ trans$$

This ensures that whenever the conclusion of a subtyping judgement is well-kinded, the types on the right- and left-hand sides of the $\leq$ will have the same kind and all subderivations will well behave. But the subtyping system is not independent from typing and kinding.

Our work started from an analysis of the algorithmic subtyping system in $\lambda P_{\leq}$ in which subtyping rules are free of kinding premises and there are no transitivity rule (which is admissible), but subtyping are defined only on β_2-normalized pretypes. Our subtyping rules can be viewed as a modification of the algorithmic subtyping rules in $\lambda P_{\leq}$ by defining the subtyping on pretypes and we have shown that the system is still well-behaved:

1. The intended subtyping rules, including reflexivity and transitivity are admissible for well-formed types.
2. The derivation tree of a well-formed subtyping judgement does not contain judgements with ill-formed types. Furthermore, typing, kinding and context formation in $\lambda\Pi_{\leq}$ are equivalent to those in $\lambda P_{\leq}$.

As in $\lambda P_{\leq}$, the type checking is decidable. But our algorithmic rules are different from those in $\lambda P_{\leq}$. First, subtyping rules in $\lambda\Pi_{\leq}$ can be directly transformed into a checking procedure, algorithmic subtyping rules are not required. Second, we do not have to do unecessary β_2-normalizations.

In $\lambda P_{\leq}$, β_2-normalizations are required for each application of the algorithmic subtyping rule:

$$\frac{\Gamma \vdash_{\mathcal{A}} M : A \quad \Gamma \vdash_{\mathcal{A}} N : B' \quad LUB_{\Gamma}(A) \equiv \pi x : B.C \quad \Gamma \vdash_{\mathcal{A}} B'^{\beta_2} \leq B}{\Gamma \vdash_{\mathcal{A}} MN : B[x := N]}$$

where LUB repeatedly β_2-normalizes a type A, replacing head variables by their bounds, see [AC96] for details. Besides, checking subtyping still needs β_2-normalization, as can be observed from the algorithmic subtyping rules in $\lambda P_{\leq}$. To compare, algorithmic application rule in $\lambda\Pi_{\leq}$ does no β_2-normalization. In subtyping algorithm, β_2-normalization is done only when needed (see rules S-ApSL, S-ApSR). Therefore, our algorithm is more efficient than that of $\lambda P_{\leq}$ since we can eliminate redundant β_2-normalizations. But $\lambda P_{\leq}$ algorithm has the advantage of low memory cost.

In [AC96], it has been conjectured that "practical implementations could make use of weak head normal forms instead of full β_2-normal forms". Our work, motivated by an attempt of basing subtyping on pretypes, has not only demonstrated the feasibility of their idea but also provided an elegant reformulation of subtyping system, which is independent from typing, free of normalizing notation, easy to extend and more efficient.

10 Future Directions

There are roughly two classes of type systems: those in which types may contain redex and those in which they do not. Accordingly, there are two classes of subtyping systems. The first class includes: subtyping dependent types, e.g. $\lambda P_{\leq}$[AC96]; subtyping F^{ω}, e.g. $F^{\omega}_{\leq}$[SP94], $F^{\omega}_{\wedge}$[Com94]. The second class includes subtyping simply typed λ-calculus, subtyping system F, e.g., $F_{\leq}$ [CG92],$F_{<:}$ [CMMS91], $Co^{\vdash}$[LMS95],[Mit88].

For the latter systems, usually, there is the property of transitivity elimination , and the subtyping is studied independently from typing system. For the former, due to the reduction relation on types, the previous approaches, e.g. $\lambda P_{\leq}$[AC96],$F^{\omega}_{\leq}$[SP94], have confined transitivity elimination on the set of (partially) normalized types, subtyping relations in these systems are strongly related to typing and kinding.

The system we have studied belongs to the first class, but we have taken a different approach than the existing ones. The subtyping has properly combined with the reduction relation so that the resulting system possesses the transitivity elimination property and we can directly work in a subtyping system separated from the typing system as one did in the second class. This makes the proofs of fundamental properties, like subject reduction, much easier and the algorithmic rules free of β_2-normalization notation.

We believe that these advantages make the system $\lambda\Pi_{\leq}$ more suitable for extensions. We want to continue the work begun here in several ways. The first is to extend dependent types with overloading and subtyping. Actually, it is the difficulty of extending $\lambda P_{\leq}$ by overloaded types that lead us to discover $\lambda\Pi_{\leq}$. In such extension, the economic logic encoding proposed by Pfenning[Pfe93] could be realized[CC96]. Second, we would like to apply the technique to other type systems in the first class. Or, since we have demonstrated that the transitivity elimination is possible for both classes of types systems, we may consider to take this approach for adding subtyping to Pure Type Systems, which covers the eight type systems in λ-cube. Third, as many authors, e.g.[Pfe93, Coq92, Luo96, AC96, Cou95, Bai96, Bar95, Sai96] , have recognised the need for integrating subtyping into type theory based proof development enviroments, like CoQ[DFH$^+$93], LEGO[LP92], Elf[Pfe89], ALF[CNSvS94],Nuprl[C$^+$86] etc, we expect that the technique developed in this report could facilitate the addition of subtyping to these systems.

11 Acknowledgements

This report was written under expert supervision of Giuseppe Longo and Giuseppe Castagna. I am grateful to Giuseppe Castagna for his careful reading of the draft version of this report and many helpful comments and suggestions. I am indebted to Giuseppe Longo, Pierre-Louis Curien for providing me the precious chance to stay at Laboratoire d'Informatique de l'École Normale Supérieure and continuously giving me encouragements and guidances in the research. I am grateful to Adriana Compagnoni for her talk in ENS and her explanation to me about $\lambda P_{\leq}$ with patience. Comments from Adriana Compagnoni and David Aspinall has improved this report.

References

[AC96] David Aspinall and Adriana Compagnoni. Subtyping dependent types, 1996. to appear in LICS96.

[Bai96] Anthony Bailey. Lego with implicit coercions, 1996. draft.

[Bar92] Henk Barendregt. Lambda calculi with types. In *Handbook of Logic in Computer Science*. Oxford University Press, 1992.

[Bar95] Gilles Barthe. Inheritance in type theory, 1995. draft.

[BM92] Kim Bruce and John Mitchell. PER models of subtyping, recursive types and higher-order polymorphism. In *Proceedings of the Nineteenth ACM Symposium on Principles of Programming Languages*, Albequerque, NM, January 1992.

[C+86] Robert L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, New Jersey, 1986.

[Car87] Luca Cardelli. Typechecking dependent types and subtypes, December 1987.

[Cas94] Giuseppe Castagna. *Surcharge, sous-typage et liaison tardive : fondements fonctionnels de la programmation orientée objets*. PhD thesis, Université Paris 7, January 1994. LIENS - Rapport de Recherche.

[CC96] Gang Chen and Giuseppe Castagna. An extension of dependent type with overloading and subtyping, 1996. Preprint.

[Che96b] Gang Chen . Subtyping Calculus of Construction – type conversion and transitivity elimination, 1996. Preprint.

[CG92] P. L. Curien and G. Ghelli. Coherence of subsumption, minimum typing and the type checking in $F_{\leq}$. *Mathematical Structures in Computer Science*, 2(1), 1992.

[CGL95] Giuseppe Castagna, Giorgio Ghelli, and Giuseppe Longo. A calculus for overloaded functions with subtyping. *Information and Computation*, 117(1):115–135, 15 February 1995. revised version of Technical Report 92-4 Laboratoire d'Informatique, Ecole Normale Supérieure - Paris.

[CMMS91] L. Cardelli, S. Martini, J.C. Mitchell, and A. Scedrov. An extension of system F with subtyping. In T. Ito and A.R. Meyer, editors, *Theoretical Aspects of Computer Software*, pages 750–771. Springer-Verlag, September 1991. LNCS 526.

[CNSvS94] Thierry Coquand, Bengt Nordström, Jan M. Smith, and Björn von Sydow. Type theory and programming. *Bulletin of the European Association for Theoretical Computer Science*, 52:203–228, February 1994.

[Com94] Adriana B. Compagnoni. Subtyping in $F^{\omega}_{\wedge}$ is decidable. Technical Report ECS-LFCS-94-281, LFCS University of Edinburgh, January 1994.

[Coq92] Thierry Coquand. Pattern matching with dependent types. In *Proceedings on Types for Proofs and Programs*, pages 71–83, 1992.

[Cou95] Judicaelë Courant. Rapport de magistère, Novembre 1995.

[DFH+93] Gilles Dowek, Amy Felty, Hugo Herbelin, Gérard Huet, Chet Murthy, Catherine Parent, Christine Paulin-Mohring, and Benjamin Werner. The Coq proof assistant user's guide. Rapport Techniques 154, INRIA, Rocquencourt, France, 1993. Version 5.8.

[Dow95] Gilles Dowek. Type theories, 1995. Notes de cours de DEA.

[HHP93] Robert Harper, Furio Honsell, and Gordon Plotkin. A framework for defining logics. *Journal of the Association for Computing Machinery*, 40(1):143–184, January 1993.

[HP92] Robert Harper and Frank Pfenning. A module system for a programming language based on the LF logical framework. Technical Report CMU-CS-92-191, Carnegie Mellon University, Pittsburgh, Pennsylvania, September 1992.

[HI92] Jason J. Hickey. Formal Abstract Data Types. Technical report, Cornell University, Computer Scicience Dependent, December,1995.

[LMS95] Giuseppe Longo, Kathleen Milsted, and Sergei Soloviev. A logic of subtyping. Technical report, Laboratoire d'Informatique, Ecole Normale Supérieure - Paris, 1995. Preliminary version in LICS'95, San Diego.

[LP92] Zhaohui Luo and Robert Pollack. The LEGO proof development system: A user's manual. Technical Report ECS-LFCS-92-211, University of Edinburgh, May 1992.

[Luo96] Zhaohui Luo. Coercive subtyping in type theory, 1996. draft.

[Mit88] J.C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76:211–249, 1988.

[Pfe89] Frank Pfenning. Elf: A language for logic definition and verified meta-programming. In *Fourth Annual Symposium on Logic in Computer Science*, pages 313–322, Pacific Grove, California, June 1989. IEEE Computer Society Press.

[Pfe93] Frank Pfenning. Refinement types for logical frameworks. In *Informal Proceedings of the 1993 Workshop on Types for Proofs and Programs*, May 1993.

[Sai96] Amokrance Saibi. Typing algorithm in type theory with inheritance, 1996. To appear in the 24th Annual SIGPLAN-SIGACT Symposium on principles of Programming Languages, Paris, France, January 15-17, 1997.

[SP94] Martin Steffen and Benjamin Pierce. Higher-order subtyping. Technical Report ECS-LFCS-94-280, LFCS University of Edinburgh, February 1994.

Appendix.

A $\lambda\Pi_{\leq}$ System

A.1 Context Formation Rules

F-empty $$\frac{}{<> \vdash \star}$$

F-term $$\frac{\Gamma \vdash A : \star \quad x \notin Dom(\Gamma)}{\Gamma, x : A \vdash \star}$$

F-type $$\frac{\Gamma \vdash K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha : K \vdash \star}$$

F-subtype $$\frac{\Gamma \vdash A : K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha \leq A : K \vdash \star}$$

F-Π $$\frac{\Gamma, x : A \vdash K}{\Gamma \vdash \Pi x{:}A.K}$$

A.2 Kinding Rules

K-var $$\frac{\Gamma \vdash \star \quad \alpha \in Dom(\Gamma)}{\Gamma \vdash \alpha : Kind_{\Gamma}(\alpha)}$$

K-π $$\frac{\Gamma, x : A \vdash B : \star}{\Gamma \vdash \pi x{:}A.B : \star}$$

K-Λ $$\frac{\Gamma, x : A \vdash B : K}{\Gamma \vdash \Lambda x{:}A.B : \Pi x{:}A.K}$$

K-app $$\frac{\Gamma \vdash A : \Pi x{:}B.K \quad \Gamma \vdash M : B}{\Gamma \vdash AM : K[x := M]}$$

K-conv $$\frac{\Gamma \vdash A : K \quad \Gamma \vdash K' \quad K =_{\beta} K'}{\Gamma \vdash A : K'}$$

A.3 Typing Rules

$$\text{T-var} \quad \frac{\Gamma \vdash \star \quad x \in Dom(\Gamma)}{\Gamma \vdash x : \Gamma(x)}$$

$$\text{T-}\lambda \quad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : \pi x{:}A.B}$$

$$\text{T-app} \quad \frac{\Gamma \vdash M : \pi x{:}A.B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x := N]}$$

$$\text{T-sub} \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash A \leq B \quad \Gamma \vdash A, B : \star}{\Gamma \vdash M : B}$$

A.4 Subtyping Rules

$$\text{S-}\pi \quad \frac{\Gamma \vdash A' \leq A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \pi x{:}A.B \leq \pi x{:}A'.B'}$$

$$\text{S-}\Lambda \quad \frac{A' =_\beta A \quad \Gamma, x : A' \vdash B \leq B'}{\Gamma \vdash \Lambda x{:}A.B \leq \Lambda x{:}A'.B'}$$

$$\text{S-ApR} \quad \frac{M_1 =_\beta M_1' \ \cdots \ M_n =_\beta M_n'}{\Gamma \vdash \alpha M_1 \cdots M_n \leq \alpha M_1' \cdots M_n'}$$

$$\text{S-ApT} \quad \frac{\Gamma \vdash \Gamma(\alpha) M_1..M_n \leq A}{\Gamma \vdash \alpha M_1..M_n \leq A}$$

$$\text{S-ApSL} \quad \frac{\Gamma \vdash B[x := M_1]M_2..M_n \leq C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq C}$$

$$\text{S-ApSR} \quad \frac{\Gamma \vdash C \leq B[x := M_1]M_2..M_n}{\Gamma \vdash C \leq (\Lambda x{:}A.B)M_1..M_n}$$

B Algorithmic Rules in $\lambda\Pi_\leq$ System

B.1 πlub Rules:

$$\text{Lub-refl} \quad \frac{}{\Gamma \vdash \pi x{:}A.B \leq_{\pi lub} \pi x{:}A.B}$$

$$\text{Lub-ApT} \quad \frac{\Gamma \vdash \Gamma(\alpha) M_1..M_n \leq_{\pi lub} A}{\Gamma \vdash \alpha M_1..M_n \leq_{\pi lub} A}$$

$$\text{Lub-ApSL} \quad \frac{\Gamma \vdash B[x := M_1]M_2..M_n \leq_{\pi lub} C}{\Gamma \vdash (\Lambda x{:}A.B)M_1..M_n \leq_{\pi lub} C}$$

B.2 Algorithmic Context Formation Rules:

AF-empty $$\frac{}{<> \vdash_{\mathcal{A}} \star}$$

AF-term $$\frac{\Gamma \vdash_{\mathcal{A}} \star \quad \Gamma \vdash_{\mathcal{A}} A : \star \quad x \notin Dom(\Gamma)}{\Gamma, x : A \vdash_{\mathcal{A}} \star}$$

AF-type $$\frac{\Gamma \vdash_{\mathcal{A}} K \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha : K \vdash_{\mathcal{A}} \star}$$

AF-subtype $$\frac{\Gamma \vdash_{\mathcal{A}} K \quad \Gamma \vdash_{\mathcal{A}} A : K' \quad K =_{\beta} K' \quad \alpha \notin Dom(\Gamma)}{\Gamma, \alpha \leq A : K \vdash_{\mathcal{A}} \star}$$

AF-Π $$\frac{\Gamma, x : A \vdash_{\mathcal{A}} K}{\Gamma \vdash_{\mathcal{A}} \Pi x{:}A.K}$$

B.3 Algorithmic Kinding Rules:

AK-var $$\frac{\alpha \in Dom(\Gamma)}{\Gamma \vdash_{\mathcal{A}} \alpha : Kind_{\Gamma}(\alpha)}$$

AK-π $$\frac{\Gamma \vdash_{\mathcal{A}} A : \star \quad \Gamma, x : A \vdash_{\mathcal{A}} B : \star}{\Gamma \vdash_{\mathcal{A}} \pi x{:}A.B : \star}$$

AK-Λ $$\frac{\Gamma \vdash_{\mathcal{A}} A : \star \quad \Gamma, x : A \vdash_{\mathcal{A}} B : K}{\Gamma \vdash_{\mathcal{A}} \Lambda x{:}A.B : \Pi x{:}A.K}$$

AK-app $$\frac{\Gamma \vdash_{\mathcal{A}} A : \Pi x{:}B.K \quad \Gamma \vdash_{\mathcal{A}} M : B' \quad \Gamma \vdash B' \leq B}{\Gamma \vdash_{\mathcal{A}} AM : K[x := M]}$$

B.4 Algorithmic Typing Rules:

AT-var $$\frac{x \in Dom(\Gamma)}{\Gamma \vdash_{\mathcal{A}} x : \Gamma(x)}$$

AT-λ $$\frac{\Gamma \vdash_{\mathcal{A}} A : \star \quad \Gamma, x : A \vdash_{\mathcal{A}} M : B}{\Gamma \vdash_{\mathcal{A}} \lambda x{:}A.M : \pi x{:}A.B}$$

AT-app $$\frac{\Gamma \vdash_{\mathcal{A}} M : A \quad \Gamma \vdash A \leq_{\pi lub} \pi x : B.C \quad \Gamma \vdash_{\mathcal{A}} N : B' \quad \Gamma \vdash B' \leq B}{\Gamma \vdash_{\mathcal{A}} MN : C[x := N]}$$