Anti-Symmetry of Higher-Order Subtyping

Adriana Compagnoni and Healfdene Goguen

1 Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030, USA, abc@cs.stevens-tech.edu, Tel:201-216-5328, Fax:201-216-8249.
2 AT&T Labs, 180 Park Ave., Florham Park, NJ 07932, USA, hhg@att.com.

Abstract. This paper shows that the subtyping relation of a higher-order lambda calculus, \( \mathcal{P}_{\xi} \), is anti-symmetric. It exhibits the first such proof, establishing in the process that the subtyping relation is a partial order—reflexive, transitive, and anti-symmetric up to \( \beta \)-equality. While a subtyping relation is reflexive and transitive by definition, anti-symmetry is a derived property. The result, which may seem obvious to the non-expert, is technically challenging, and had been an open problem for almost a decade. In this context, typed operational semantics for subtyping offers a powerful new technology to solve the problem: of particular importance is our extended rule for the well-formedness of types with head variables. The paper also gives a presentation of \( \mathcal{P}_{\xi} \) without a relation for \( \beta \)-equality, apparently the first such, and shows its equivalence with the traditional presentation.

1 Introduction

Object-oriented programming languages such as Smalltalk, C++, Modula 3, and Java have become popular because they encourage and facilitate software reuse and abstract design. One attempt to give a theoretical understanding of these object-oriented programming languages has been to introduce type systems with features to model constructs from object-oriented programming languages [8, 10], for example bounded quantification [20] and recursive types [2].

Metatheoretic properties of the type systems are important to justify the programming languages being modeled. One important property of a type system is subject reduction or type preservation, which states that evaluation of programs preserves their type. This is one of the central results of an earlier paper [15] about \( \mathcal{P}_{\xi} \), which also showed the correctness of the algorithms for type-formation and subtyping. Another important property for type systems is the decidability of type-checking and subtyping: a compiler should be able to process basic type information reliably without help from the programmer in order to prevent basic programming errors. Decidability of type-checking ensures that this will always be possible, and decidability of subtyping is a crucial step to proving this. This result was proved for our calculus in [14].

* Most of this author's work was carried out at LFCS, University of Edinburgh, JCM, The King's Buildings, Edinburgh, EH9 3JZ, UK.
The subtyping relation has been extensively researched because of its importance in applications to programming languages [1, 6, 21, 28], proof assistants [3, 25, 29], and metatheoretical studies [4, 5, 7, 10, 14–16, 18, 19, 24, 30], to name a few. However, none of these studies has established the anti-symmetry of the subtyping relation for a higher-order calculus. In some cases it has been conjectured, as in [35]. In other cases, the problem is avoided by taking an equality that satisfies anti-symmetry by definition: \( A = B \) is defined as \( A \leq B \) and \( B \leq A \). Steffen [33] has shown the simpler property, appropriate to his setting of polarized subtyping, that \( A \leq B \) and \( B \leq A \), where both derivations are without uses of promotion\(^1\), if and only if \( A =_\rho B \). However, in most higher-order subtyping calculi, including ours, the problem in showing anti-symmetry is exactly to show that the derivations of \( A \leq B \) and \( B \leq A \) contain no uses of promotion.

Anti-symmetry has been demonstrated for \( F_\leq \)[20]. For Mitchell’s second order \( \lambda \)-calculus a la Curry with subtyping, a completely different style of subtyping from the one we consider here, anti-symmetry has been studied under the name of equational axiomatization of biocoercibility [34]. There, \( A \) and \( B \) are called biocoercible if \( A \leq B \) and \( B \leq A \), and the paper proves that if \( A \) and \( B \) are biocoercible then \( A \equiv B \), for an appropriate equivalence relation \( \equiv \). However, the problem of biocoercibility of \( F_\leq \) is considerably easier than anti-symmetry of higher-order subtyping, because there is no notion of computation on types, and in particular no \( \beta \)-reduction on types.

The rest of the paper is structured as follows. In the remaining sections of the introduction we discuss technical points relating to typed operational semantics and anti-symmetry of subtyping. In Section 2 we introduce the basic language of \( \mathcal{F}_\leq \). In Section 3 we introduce the typed operational semantics, which as explained above plays a central role in our proof of anti-symmetry by providing a powerful induction principle. Section 4 outlines the proof of anti-symmetry using the typed operational semantics and sketches an approach to implementing subtyping and equality simultaneously. In Section 5 we give a sketch of the metatheory of a new presentation of \( \mathcal{F}_\leq \) without judgemental equality or conversion. Finally, we draw conclusions in Section 6. The appendices contain an outline of the results that we use from our earlier development of typed operational semantics for \( \mathcal{F}_\leq \), rules for the traditional presentation of \( \mathcal{F}_\leq \), and rules for the typed operational semantics.

**Typed Operational Semantics** Our proof of anti-symmetry of higher-order subtyping relies on our understanding of subtyping built up using typed operational semantics. Typed operational semantics [22] gives an alternative induction principle for type theories, by presenting type theory operationally rather than declaratively. The typed operational semantics for \( \mathcal{F}_\leq \) has judgements for reduction to weak-head and normal form for types, and for subtyping comparison between types in weak-head normal form and arbitrary types. Because the system is presented from the perspective of computation, many properties about

\(^1\) Promotion is a step of transitivity along the bound of the head variable, and consists of replacing the head variable by its bound in a given context.
the relationship between reduction and typing are particularly easy to show by induction on derivations of the typed operational semantics. This topic is dealt with extensively in an earlier article [14].

A typed approach like typed operational semantics is essential to studying anti-symmetry. Similar to Church–Rosser for \( \beta\eta \)-reduction in type theories, which is only true for typed terms, anti-symmetry is only true for well-formed judgements. For example, in \( Y \subseteq (AX \subseteq K.\, (X \circ X)) : K' \), \( Z \subseteq Y : K'' \), an invalid context, we can show that \( Z \subseteq Y \) and \( Y \subseteq Z \), but the two types are clearly not \( \beta \)-equal.

A key property in our proof of anti-symmetry, that \( \Gamma(X)A_{1}, \ldots , A_{n} \) \( \subseteq X_{1}(A_{1}, \ldots , A_{n}) \) is undervariable in the context \( \Gamma \) (where \( \Gamma(X) \) denotes the bound of \( X \) in \( \Gamma \)), also relies on the well-formedness of the judgement. Intuitively, if \( \Gamma(X)A_{1}, \ldots , A_{n} \) and \( X_{1}(A_{1}, \ldots , A_{n}) \) are well-formed then they can never be \( \beta \)-equal. While the base case, that \( \Gamma(X) \) \( \subseteq X \) is impossible, is straightforward, the complication in the general case is that \( X \) may appear in some \( A_{i} \), so it is not obvious that \( \Gamma(X)A_{1}, \ldots , A_{n} \) may not \( \beta \)-reduce to \( X_{1}(A_{1}, \ldots , A_{n}) \), for example. The type system rules out cases like \( \Gamma(X) \) being \( AY \subseteq A : K(Y \circ Y) \) and \( A_{1} \) being \( X \).

Essential to our approach is the extended rule \( \text{ST-T} \text{Av} \) from our most recent papers on decidability of subtyping [14, 15]. This rule contains as a premise the well-formedness of the bound \( \Gamma(X) \) applied to a sequence of types, in order to conclude the well-formedness of the variable \( X \) applied to that sequence of terms. This rule is justified by an extension of the logical relation proof of Soundness (Theorem 3) using saturated sets. It is the powerful induction principle arising out of this rule, already crucial to our proof of decidability, that allows us to show the undervariableness of \( \Gamma(X)A_{1}, \ldots , A_{n} \) \( \subseteq X_{1}(A_{1}, \ldots , A_{n}) \).

**Induced Equivalence Relations** The equivalence relation induced by \( A \leq B \) and \( B \leq A \) may be stronger than the usual intensional equality associated with type theory, syntactic equivalence on normal forms. One such case occurs in [32], where the types \( \forall(X <: \text{Bot})X \rightarrow X \) and \( \forall(X <: \text{Bot})\text{Bot} \rightarrow \text{Bot} \) are “equivalent in the subtype relation,” even though they are not syntactically identical. A similar situation appears in intersection types disciplines, where \( \top = A \rightarrow \top \) and also \( A \rightarrow \top \leq \top \) and \( \top \leq A \rightarrow \top \). A final example is extensible records [11], where the extension operator is associative and commutative in the subtyping relation. These equivalence relations have models in existing frameworks, for example game semantics [13] or PER models [17].

Such equivalences also arise in the context of programming languages. For example, consider object types with a private section and a public interface, where two object types \( O_{1} \) and \( O_{2} \) may satisfy \( O_{1} \leq O_{2} \) and \( O_{2} \leq O_{1} \) but differ in their private section. The equivalence relation that only considers the public interface will be more useful to the programmer than intensional equality, because it is more permissive.

\(^2\) \( A \) and \( B \) are equivalent in the subtyping relation if \( A \leq B \) and \( B \leq A \).
In our current work on $\mathcal{F}_{\leq}$, the equivalence relation is the usual notion of $\beta$-equality at the type level. However, Martin-Löf [26] has demonstrated how to capture more sophisticated equivalence relations in intensional type theory. He allows the equality on elements of a type to include an arbitrary decidable equivalence relation on the normal forms of a type, rather than taking simple syntactic equivalence of normal forms. To extend our work to the example above of public and private sections of an object type, we can take an equivalence on object types in normal form that simply compares the fields of the public interfaces.

**Metatheoretic Consequences of Anti-Symmetry** In addition to providing the answer to a long-standing open problem, our work has several consequences in the development of the metatheory of type systems with subtyping. First, in the presence of anti-symmetry for subtyping, we can show the equivalence between traditional presentations, either with judgemental equality or untyped $\beta$-conversion, and a system without the notion of equality. This means that the proof of soundness for model constructions, such as that for proofs of strong normalization or PER models, can be more concise. In Section 5 we give a sketch of a new presentation of $\mathcal{F}_{\leq}$ without judgemental equality or conversion, apparently not found elsewhere in the literature, and discuss how the development of the metatheory proceeds for this system. The lack of a notion of equality may also have consequences for the implementation of type theories with subtyping.

Another consequence of this result is that we can now prove the Minimum Types Property, as opposed to the Minimal Types Property normally proved. A type inference algorithm can be shown to find one of many minimal types for a term. We can now clearly state the relationship between all of these minimal types: whereas before we knew that any two minimal types were subtypes of each other, we now know that they are $\beta$-equal.

## 2 Syntax

We now introduce the basic language of $\mathcal{F}_{\leq}$. A complete development of its meta-theory can be found in [15].

The kinds of $\mathcal{F}_{\leq}$ are the kind $\star$ of proper types and the kinds $\Pi X \leq A; K_1, K_2$ of functions on types, or type operators. The types of $\mathcal{F}_{\leq}$ are a straightforward higher-order extension of $F_{\leq}$, where we allow bounds on the abstraction $\Lambda X \leq A; K_1, B$. There is a top type $T_\star$, and we define top type operators $T_K$ at every kind $K$ by $T_{\Pi X \leq A_1; K_1, K_2} = \Lambda X \leq A_1; K_1, T_{K_2}$. The language of terms is the same as that for $F_{\leq}$, with bounded type abstraction $\Lambda X \leq A; K, M$. As in $F_{\leq}$, each type variable is given an upper bound at the point where it is introduced.

The operational semantics of $\mathcal{F}_{\leq}$ is given by the usual $\beta$-reduction rules on terms and types, and is extended to a compatible relation with respect to term or type formation. We write $\rightarrow_{\beta}$ for the transitive and reflexive closure of $\rightarrow_{\beta}$, and $=_{\beta}$ for the least equivalence relation containing $\rightarrow_{\beta}$ and closed under $\alpha$-
equivalence. We write $A^nf$ to indicate the $\beta$-normal form of $A$, and $A \equiv B$ when $A$ and $B$ are $\alpha$-equivalent.

Weak head reduction, or leftmost outermost reduction, is a less familiar notion from lambda calculus that appears in our presentation.

For technical reasons relating to the model construction, we need to consider a slightly stronger notion of weak-head normal form.

**Definition 1 (Weak-Head Normal).** The types $T_x$, $A_1 \rightarrow A_2$, $\forall X \leq A : K.B$, and $A \forall X \leq A : K.B$ are weak-head normal. $X(A_1, \ldots, A_m)$ is weak-head normal if $A_1, \ldots, A_m$ are in normal form.

Contexts are defined as usual, where the empty context is written $\emptyset$, term variable bindings have the form $x : A$, and type variable bindings have the form $X \leq A : K$. We write $\text{dom}(\Gamma)$ for the set of term and type variables defined in a context $\Gamma$. The sets of free term and type variables occurring in terms, types, kinds, contexts or judgements are written $\text{FV}(\_)$ and $\text{FTV}(\_)$.

Since we are careful to ensure that no variable is bound more than once, we sometimes consider contexts as finite functions: $\Gamma(X)$ yields the bound of $X$ in $\Gamma$, where $X \in \text{dom}(\Gamma)$ is implicitly asserted.

We write $A(B_1, \ldots, B_n)$ for $((A B_1) \ldots B_n)$. If $A$ is of the form $X(B_1, \ldots, B_n)$ then $A$ has head variable $X$. We write $\text{HV}(\_)$ for the partial function returning the head variable of a term. We write $B[X \leftarrow A]$ for the capture-avoiding substitution of $A$ for $X$ in $B$. We identify types that differ only in the names of bound variables.

The system $\mathcal{F}_\leq$ is presented as simultaneously defined inductive relations with the following judgement forms:

- $\Gamma \vdash \text{ok}$ well-formed context
- $\Gamma \vdash K$ well-formed kind
- $\Gamma \vdash K \equiv_{\beta} K'$ kind equality
- $\Gamma \vdash A : K$ well-kind type
- $\Gamma \vdash A =_{\beta} B : K$ type equality
- $\Gamma \vdash A \leq B : K$ subtype
- $\Gamma \vdash M : A$ well-typed term.

We sometimes use the metavariable $J$ to range over statements (right-hand sides of judgements) of any of these judgement forms.

We now give an overview of the rules of inference for $\mathcal{F}_\leq$. The context formation rules are as usual in $\mathcal{F}_\leq$. Kind formation differs by incorporating information about the bounds in $\Pi$:

$$
\Gamma, X \leq A : K_1 \vdash K_2 \\
\Gamma \vdash \Pi X \leq A : K_1, K_2
$$

\text{(K-H)}

The rules of inference for kind equality simply define a typed equivalence relation compatible with respect to the kind formers.

The rules for type formation similarly need to be adjusted for bounded operator abstraction.

$$
\Gamma, X \leq A_1 : K_1 \vdash A_2 : K_2 \\
\Gamma \vdash \lambda X \leq A_1 : K_1, A_2 : \Pi X \leq A_1 : K_1, K_2
$$

\text{(T-TAna)}

$$
\Gamma \vdash A : \Pi X \leq B : K_1, K_2 \\
\Gamma \vdash C \leq B : K_1 \\
\Gamma \vdash A C : K_2[X \leftarrow C]
$$

\text{(T-TAa)}
\[ \Gamma \vdash A : K \quad \Gamma \vdash K \equiv_{\beta} K' \quad \Gamma \vdash A : K' \]  

(\text{T-Conv})

Type equality is defined as the typed equivalence relation compatible with respect to the type formers and closed under \(\beta\)-reduction for types.

\[ \Gamma, X \leq A_1 ; K_1 \vdash A_2 : K_2 \quad \Gamma \vdash C \leq A_1 : K_1 \]  

\[ \Gamma \vdash (AX \leq A_1 ; K_1 , A_2) C \equiv_{\beta} A_2 [X \leftarrow C] : K_2 [X \leftarrow C] \]  

(\text{T-Eq-Beta})

The type equality rules appear in Appendix B.

The subtyping rules are again those of \( F^2 \) [9, 10, 27], except for those dealing with bounded type abstraction and type application and the rule for subtyping the quantifier. We chose Cardelli and Wegner’s kernel Fun rule for quantifiers with equal bounds [12], because the contravariant rule for quantifiers renders the system undecidable [31]. Furthermore, transitivity elimination in the presence of such a rule in the higher-order case remains an open problem. Type equality is included in subtyping.

\[ \Gamma \vdash A =_{\beta} B : K \]  

\[ \Gamma \vdash A \leq B : K \]  

(S-Conv)

The subtyping rules also appear in Appendix B. Our goal is to prove that this relation is anti-symmetric up to \(\beta\)-equality.

The term formation rules are standard. We remind the reader that the subtyping relation on \(*\) induces an inclusion over types.

\[ \Gamma \vdash M : A \quad \Gamma \vdash A \leq B : * \]  

\[ \Gamma \vdash M : B \]  

(\(\tau\)-Sum)

3 The Typed Operational Semantics

The typed operational semantics for \( F^2 \) is organized in five judgement forms:

\[ \Gamma \vdash S \text{ ok} \quad \text{valid context} \]  

\[ \Gamma \vdash S \; A \rightarrow_{w} B \rightarrow_{n} C : K \quad \text{type reduction} \]  

\[ \Gamma \vdash S \; K \rightarrow_{n} K' \quad \text{kind reduction} \]  

\[ \Gamma \vdash S \; A \leq_{W} B : K \quad \text{subtyping} \]  

\[ \Gamma \vdash S \; A \leq_{W} B : K \quad \text{weak-head subtyping} \]  

This system can be understood informally as a particularly informative algorithm for kinding and subtyping in \( F^2 \). The first judgement simply represents well-formed contexts. The second and third judgements represent normalization of kinds and types, where the formulation of the rules of inference ensures strong normalization, Church–Rosser and other good metatheoretic properties. The last two judgements represent an algorithm for subtyping that first reduces the left and right-hand sides to weak-head normal forms and then compares these.

We use various notations that omit unnecessary components of the judgement. For example, we write \( \Gamma \vdash S \; K \) for \( \Gamma \vdash S \; K \rightarrow_{n} K' \) for some \( K' \), and similarly for types, \( \Gamma \vdash S \; A \rightarrow_{w} B : K \) for \( \Gamma \vdash S \; A \rightarrow_{w} B \rightarrow_{n} C : K \) for some \( C \), and \( \Gamma \vdash S \; A \rightarrow_{w} B : K \) for \( \Gamma \vdash S \; A \rightarrow_{w} B \rightarrow_{n} C : K \). We also write \( \Gamma \vdash S \; K , K' \rightarrow_{n} K'' \) when \( \Gamma \vdash S \; K \rightarrow_{n} K'' \) and \( \Gamma \vdash S \; K' \rightarrow_{n} K'' \), and similarly for types.
We now present the rules of inference. The context formation and kind normalization rules are modifications of the corresponding rules for $F^\kappa$. The rules for type reduction combine kinding information and computational behavior in the form of weak-head and $\beta$-normal forms. For example, the rule for arrow types says how to obtain the weak-head and $\beta$-normal forms of $A_1 \rightarrow A_2$ in $\star$ from those for $A_1$ and $A_2$ in $\star$.

Kind reduction depends on reduction on types. This is not the case in Cardelli's $F^\kappa$, or in $\mathcal{O}b_{\omega_1}[1]$, where kinds are syntactically simpler.

\[
\Gamma \vdash_S K_1 \rightarrow_n K_1' \quad \Gamma \vdash_S A \rightarrow_n B : K_1' \quad \Gamma, X \leq_A : K_1 \vdash_S K_2 \rightarrow_n K_2' \quad \frac{}{\Gamma \vdash_S \Pi X \leq_A : K_1, K_2 \rightarrow_n \Pi X \leq_B : K_1', K_2'} \quad (SK-\Pi)
\]

The rules for type reduction are in Appendix C.

The beta rule, besides uncovering the outermost redex of the application $BC$ and contracting it, finds the weak-head normal form $E$ and the normal form $F$. The premise $\Gamma \vdash_S K_2[X \leftarrow C] \rightarrow_w K$ ensures that $E$ and $F$ have $\beta$-equal kinds, and the subtyping premise $\Gamma \vdash_S C \leq A : K_1'$ enforces the well-formation of $BC$.

\[
\Gamma \vdash_S B \rightarrow_n A \quad \Pi X \leq A : K_1, D : \Pi X \leq A' : K_1', K_2 \quad \Gamma \vdash_S K_2[X \leftarrow C] \rightarrow_n K \quad \frac{}{\Gamma \vdash_S \Pi X \leq A : K_1, K_2 \rightarrow_n \Pi X \leq B : K_1', K_2'} \quad (\text{ST-Beta})
\]

The weak-head subtyping rules are motivated by the algorithmic rules in [16]. The rules SWS-Arrow, SWS-All, and SWS-TApp are structural. The rule SWS-TApp implicitly uses transitivity, reducing the problem of a variable being less than another type to the problem of the bound of the variable being less than that type. The side condition ensures determinism. The rules for weak-head subtyping are in Appendix C.

Finally, full subtyping is defined by reference to the weak-head subtyping relation.

\[
\Gamma \vdash_S A \rightarrow_w C : K \quad \Gamma \vdash_S B \rightarrow_w D : K \quad \Gamma \vdash_S C \leq_W D : K \quad \frac{}{\Gamma \vdash_S A \leq B : K} \quad (\text{SS-Tac})
\]

We have developed extensive results for this system in [15]. Those relevant to our development here, including the equivalence of the original system and the typed operational semantics, are summarized in Appendix A.

4 Anti-Symmetry

Our goal is to prove that if $\Gamma \vdash A \leq B : K$ and $\Gamma \vdash B \leq A : K$ then $\Gamma \vdash A =_\beta B : K$.

We obtain this (Proposition 7) as a consequence of the corresponding property in the typed operational semantics using Soundness (Theorem 3) and Completeness (Proposition 2). In the semantics, the way to say that two types are equal is that they have the same normal form.

Our choice of terminology for Soundness and Completeness comes from proofs of strong normalization using saturated sets, where the Soundness theorem says that everything derivable in the syntax is satisfied in the saturated sets model.
The main difficulty appears when $A$ is of the form $X(A_1, \ldots, A_m)$. Intuitively, in this case $B$ can only be $X(A_1, \ldots, A_m)$. To discard the possibility that $B$ could be other than $A$, we need to prove that $\Gamma \vdash S \Gamma(X)(A_1, \ldots, A_m) \leq X(A_1, \ldots, A_m) : K$ is impossible.

4.1 Basic Properties

In this section we establish some preliminary lemmas.

The typed operational semantics is deterministic (Lemma 13), so there is only one derivation for each judgement, and we can invert the rules deriving premises from conclusions. We sometimes use this fact without mentioning it.

**Lemma 1.** Suppose $\Gamma \vdash S A \rightarrow B : \Pi X \leq E : K_1, K_2$, $\Gamma \vdash S C \leq E : K_1$ and $\Gamma \vdash S K_2[X \leftarrow C] \rightarrow_n K_2'$. Then there is an $F$ such that $\Gamma \vdash S AC \rightarrow_n F : K_2'$

and $\Gamma \vdash S BC \rightarrow_n F : K_2'$.

**Proof.** By Completeness, equational reasoning and Soundness.

As shown in [15], if $\Gamma \vdash S A \rightarrow^w B \rightarrow^w C : K$ then $B$ is the weak-head normal form of $A$ and $C$ the normal form of $A$. Therefore, the following structural property of the typed operational semantics can be read as follows: if $B$ is the weak-head normal form of $A$, then $B$ is itself weak-head normal, and if $C$ is the normal form of $A$, then $C$ has itself as weak-head and normal forms.

**Lemma 2.**

1. If $\Gamma \vdash S A \rightarrow^w B \rightarrow^w C : K$ then $\Gamma \vdash S B \rightarrow^w B \rightarrow^w C : K$ and $\Gamma \vdash S C \rightarrow^w C \rightarrow^w C : K$.
2. If $\Gamma \vdash S K \rightarrow_n K'$ then $\Gamma \vdash S K' \rightarrow_n K'$.

**Proof.** By simultaneous induction on derivations, where most cases are immediate or follow by the induction hypothesis. The interesting case, **ST-TApp**, follows by the induction hypothesis and Subtyping Conversion.

**Lemma 3 (Upper Bound).**

1. If $\Gamma \vdash S X(A_1, \ldots, A_m) : K$ then $\Gamma \vdash S \Gamma(X)(A_1, \ldots, A_m) : K$.
2. If $\Gamma \vdash S X(A_1, \ldots, A_m) \leq B : K$ and $B \not\equiv X(A_1, \ldots, A_m)$ then $\Gamma \vdash S$ $\Gamma(X)(A_1, \ldots, A_m) \leq B : K$.
3. If $\Gamma \vdash S X(A_1, \ldots, A_m) \leq^w B : K$ and $B \not\equiv X(A_1, \ldots, A_m)$ then $\Gamma \vdash S$ $\Gamma(X)(A_1, \ldots, A_m) \leq B : K$.

**Proof.** Case 1 follows by Soundness and Completeness from Lemma 16, because $\Gamma \vdash S K \rightarrow_n K$. Cases 2 and 3 follow by simultaneous induction on derivations.
4.2 An Impossible Judgement

In this section we show that \( \Gamma \vdash S \Gamma(X)(A_1 \ldots A_m) \leq X(A_1 \ldots A_m) : K \) is an underviable judgement.

The particular property that drives our proof of this result is that the typing rule \(ST\text{-}\text{TAPP}\) in the typed operational semantics, for variables applied to a sequence of types, has a subderivation stating the well-formedness of the type resulting from the replacement of the variable with its bound. This rule of inference is justified by our earlier proofs of Soundness and Completeness of the usual rules of inference for the typed operational semantics, published elsewhere [15].

The development of the full metatheory of typed operational semantics is quite complex, and in particular the proof of Soundness is similar to proofs of strong normalization and relies on logical relations. However, once we have established the equivalence of the original presentation and the typed operational semantics, we can define a measure, from derivations to natural numbers, that counts the number of uses of promotion (replacing a variable by its bound in the operational semantics) before reaching \(\text{Top}\). Clearly, the number of uses of the rule \(SW\text{-}\text{TAPP}\) in a derivation of \(\Gamma \vdash S \Gamma(X)(A_1, \ldots, A_n) \leq T : K\) must be greater than the number of uses in a derivation of \(\Gamma \vdash S \Gamma(X)(A_1, \ldots, A_n) \leq T : K\), because the latter is a subderivation of the former. On the other hand, we can also show that if \(\Gamma \vdash S A \leq B : K\) then the number of uses of the rule \(SW\text{-}\text{TAPP}\) in \(\Gamma \vdash S A \leq T : K\) is greater than or equal to the number in \(\Gamma \vdash S B \leq T : K\). Hence, if we have a derivation of \(\Gamma \vdash S \Gamma(X)(A_1, \ldots, A_n) \leq X(A_1, \ldots, A_n) : K\) then we can derive a contradiction.

We introduce a function from derivations in the typed operational semantics to numbers, informally capturing the number of uses of the rule of inference that a variable is less than its bound. We do not have a good notation for defining a function on derivations because such definitions do not occur commonly in the literature. We therefore abbreviate in the following definition \(\#(X)\) for the subderivation of the bound of \(X\) in \(\Gamma\) for the case \(ST\text{-}\text{TVAR}\); \(\#(X)\text{nf}(A_1, \ldots, A_m)\) for the subderivation of the normal form of the bound applied to the arguments of \(X\) for the case \(ST\text{-}\text{TAPP}\); and \(\beta\) for a derivation of \(BC\) using \(ST\text{-}\text{Beta}\) and \(\text{reduct}\) for the subderivation of the \(\beta\)-reduct of \(BC\).

**Definition 2.** We define \(\#(-)\), from derivations of \(\Gamma \vdash S A \rightarrow_{w} B \rightarrow_{n} C : K\) to numbers, by induction on derivations:

\[
\begin{align*}
\text{ST-Top} \quad &\#(T_v) = 0 & \text{ST-TAPP} \quad &\#(X(A_1, \ldots, A_m)) = \\
\text{ST-Arrow} \quad &\#(A_1 \rightarrow A_2) = 0 & \#(\Gamma(X)\text{nf}(A_1, \ldots, A_m)) + 1 \\
\text{ST-All} \quad &\#(\forall A_1 : K.A_2) = 0 & \text{ST-TVAR} \quad &\#(X) = \#(\Gamma(X)) + 1 \\
\text{ST-TARS} \quad &\#(A_1 \leq : A_1 : K.A_2) = 0 & \text{ST-Beta} \quad &\#(\beta) = \#(\text{reduct})
\end{align*}
\]

Notice that if \(\Gamma \vdash S A \leq B : K\) then \(\Gamma \vdash S A : K\) and \(\Gamma \vdash S B : K\) by Lemma 17, so \(\#(A)\) and \(\#(B)\) are defined.

We now show that this length function is invariant with respect to well-formed types or type-operators that have the same normal form. This lemma
justifies our later informality in writing types or type-operators in place of their derivations of well-formedness.

**Lemma 4.** If $\mathcal{D}$ is a derivation of $\Gamma \vdash S A \rightarrow_{w} C \rightarrow_{w} n E : K$ and $\mathcal{D}'$ is a derivation of $\Gamma \vdash S B \rightarrow_{w} D \rightarrow_{w} n E : K$ then $\sharp(\mathcal{D}) = \sharp(\mathcal{D}')$.

**Proof.** By induction on derivations $\mathcal{D}$.

In each case where $E$ is a type constructor or a variable, we perform a nested induction on $\mathcal{D}'$, and there are two interesting cases. The first is when the nested case is the same rule as for the outermost induction. In this case the lengths are equal by assumption for rules $\text{ST-Top}$, $\text{ST-Brow}$, $\text{ST-All}$ and $\text{ST-Tabs}$. For $\text{ST-Var}$ and $\text{ST-App}$, we know $\Gamma(X)$ is a function, so the result follows by Determinacy and the induction hypothesis. In the case $\text{ST-Tab}$ we have that $\Gamma \vdash S A \rightarrow_{w} A_{1} A_{2} \rightarrow_{w} X(A_{1}, \ldots, A_{m}, F) \rightarrow_{w} X(A_{1}, \ldots, A_{m}, F)$: $K$ and $\Gamma \vdash S B_{1} B_{2} \rightarrow_{w} X(A_{1}, \ldots, A_{m}, F) \rightarrow_{w} X(A_{1}, \ldots, A_{m}, F)$: $K$. Then $\mathcal{D}$ and $\mathcal{D}'$ have the same subderivation $\Gamma \vdash S D(A_{1}, \ldots, A_{m}, F)$: $K$. Hence, by the definition of $\sharp(-)$, $\sharp(\mathcal{D}) = \sharp(D(A_{1}, \ldots, A_{m})) + 1 = \sharp(\mathcal{D}')$.

The second interesting case in the nested inductions is $\text{ST-Beta}$, where the result follows by the induction hypothesis. All of the other nested cases contradict the assumption that the normal forms are the same.

The final outermost case, $\text{ST-Beta}$, follows by the induction hypothesis. \(\square\)

Then, if $\Gamma \vdash S X(A_{1}, \ldots, A_{m}) : K$ we have

$$
\sharp(X(A_{1}, \ldots, A_{m})) = \sharp(\Gamma(X) \text{cd}(A_{1}, \ldots, A_{m})) + 1
= \sharp(\Gamma(X)(A_{1}, \ldots, A_{m})) + 1 > \sharp(\Gamma(X)(A_{1}, \ldots, A_{m}))
$$

where by Lemma 1 there is a $B$ such that $\Gamma \vdash S \Gamma(X) \text{cd}(A_{1}, \ldots, A_{m}) \rightarrow_{w} n B : K$ and $\Gamma \vdash S \Gamma(X)(A_{1}, \ldots, A_{m}) \rightarrow_{w} n B : K$, and by Lemma 4 they have the same length.

Now, we come to the main lemma about $\sharp(A)$:

**Lemma 5.**

1. If $\Gamma \vdash S A \leq B : K$ then $\sharp(A) \geq \sharp(B)$.
2. If $\Gamma \vdash S A \leq_{w} B : K$ then $\sharp(A) \geq \sharp(B)$.

**Proof.** By simultaneous induction on derivations. In Case 1, the only possible rule for $\Gamma \vdash S A \leq B : K$ is $\text{SS-Asc}$, which follows by Determinacy, Lemma 4, and the induction hypothesis. In Case 2 the only interesting case is $\text{SWS-App}$, where by the induction hypothesis $\sharp(E) \geq \sharp(A)$, so

$$
\sharp(\Gamma((A_{1}, \ldots, A_{m}))) = \sharp(\Gamma(X) \text{cd}(A_{1}, \ldots, A_{m})) + 1 = \sharp(E) + 1 > \sharp(E) \geq \sharp(A),
$$

where we know that $\sharp(\Gamma(X) \text{cd}(A_{1}, \ldots, A_{m})) = \sharp(E)$ by Determinacy. \(\square\)

**Lemma 6.** There can be no derivation of the judgement

$$
\Gamma \vdash S \Gamma(X)(A_{1}, \ldots, A_{m}) \leq X(A_{1}, \ldots, A_{m}) : K.
$$
\textbf{Proof.} Suppose that there were a derivation of $\Gamma \vdash_S \Gamma(X)(A_1, \ldots, A_m) \leq X(A_1, \ldots, A_m) : K$. By Lemma 5 we have
\[ \#(\Gamma(X)(A_1, \ldots, A_m)) \geq \#(X(A_1, \ldots, A_m)) \], contradicting the fact that
\[ \#(X(A_1, \ldots, A_m)) > \#(\Gamma(X)(A_1, \ldots, A_m)) \]. \hfill \Box

Based on our understanding of the behavior of bounds in Lemma 3, the negative result Lemma 6 seems intuitive and believable. However, the results in Lemma 3 have been known for systems of higher-order subtyping, while Lemma 6 has remained a conjecture, so this was the major challenge of our development.

4.3 Main Result

We can now prove our main result. Observe that in the case \textsc{sws-tapp} we use our key lemma (Lemma 6).

\textbf{Lemma 7 (Anti-Symmetry of TOS).}

1. If $\Gamma \vdash_S A \leq B : K$ and $\Gamma \vdash_S B \leq A : K$ then $\Gamma \vdash_S A, B \rightarrow_n C : K$.
2. If $\Gamma \vdash_S A \leq_w B : K$ and $\Gamma \vdash_S B \leq_w A : K$ then $\Gamma \vdash_S A, B \rightarrow_n C : K$.

\textbf{Proof.} The argument is by simultaneous induction on derivations.

1. The only rule to derive both assumptions is \textsc{ss-bc}. By Determinacy, the premises are $\Gamma \vdash_S A \rightarrow_w C \rightarrow_n \rightarrow_n E : K$, $\Gamma \vdash_S B \rightarrow_w D \rightarrow_n \rightarrow_n F : K$, $\Gamma \vdash_S C \leq_w D : K$, and $\Gamma \vdash_S D \leq_w C : K$. By Lemma 2 $\Gamma \vdash_S C \rightarrow_w C \rightarrow_n \rightarrow_n E : K$ and $\Gamma \vdash_S D \rightarrow_w D \rightarrow_n \rightarrow_n F : K$. By the induction hypothesis $\Gamma \vdash_S C \rightarrow_n G : K$ and $\Gamma \vdash_S D \rightarrow_n G : K$. Finally, $E \equiv F \equiv G$ by Determinacy.

2. The proof is by induction on the derivation of $\Gamma \vdash_S A \leq_w B : K$.

\textsc{sws-top} Then $B \equiv T_*$, and $HV(A)$ undefined, which means that $A$ is not a type application or a type variable. Therefore the only rule to derive $\Gamma \vdash_S T_* \leq_w A : K$ is \textsc{sws-top}, which means that $A \equiv T_*$ By Lemma 17 and Determinacy, $\Gamma \vdash_S T_* \rightarrow_w T_* \rightarrow_n T_* : *$.

\textsc{sws-tapp} We are going to show that this case is not possible. Assume that it is. We have that $A \equiv X(A_1, \ldots, A_m)$ and $B \equiv X(A_1, \ldots, A_m)$. By Upper Bound (Lemma 3), $\Gamma \vdash_S \Gamma(X)(A_1, \ldots, A_m) \leq B : K$, and by transitivity $\Gamma \vdash_S \Gamma(X)(A_1, \ldots, A_m) \leq X(A_1, \ldots, A_m) : K$, which is a contradiction, by Lemma 6.

\textsc{sws-rwll} By the premise.

\textsc{sws-arrow} The only rule to derive $\Gamma \vdash_S B \leq_w A : K$ is also \textsc{sws-arrow}, so the result follows by the induction hypothesis and \textsc{st-arrow}.

\textsc{sws-all} The only rule to derive $\Gamma \vdash_S B \leq_w A : K$ is also \textsc{sws-all}. The premises are, $\Gamma, X \leq A_1 : K \vdash_S A_2 \leq \leq B : X \leq B_1 : K_1 \vdash_S B_2 \leq \leq A_2 : X, \Gamma \vdash_S A_1, B_1 \rightarrow_n C_1 : K''$, and $\Gamma \vdash_S K, K' \rightarrow_n K''$. By Context Conversion (Lemma 12), $\Gamma, X \leq A_1 : K \vdash_S B_2 \leq \leq A_2 : X$. We can now apply the induction hypothesis to obtain $\Gamma, X \leq A_1 : K \vdash_S A_2, B_2 \rightarrow_n C_2 : X$. Finally, $\Gamma \vdash_S (\forall X \leq A_1 : K, A_2), (\forall X \leq B_1 : K', B_2) \rightarrow_n (\forall X \leq C_1 : K'', C_2) : X$, by \textsc{st-all}. 

\text{end of page}
Theorem 1 (Anti-Symmetry). If $\Gamma \vdash A \leq B : K$ and $\Gamma \vdash B \leq A : K$ then $\Gamma \vdash A =_\beta B : K$.

Proof. By Soundness (Theorem 3), $\Gamma \vdash_S A \leq B : K'$, $\Gamma \vdash_S B \leq A : K'$, and $\Gamma \vdash_S K \Rightarrow_n K'$. By Proposition 7, $\Gamma \vdash_S A, B \Rightarrow_n C : K'$. By Completeness and symmetry of kind equality $\Gamma \vdash K' =_\beta K$. By Completeness $\Gamma \vdash A =_\beta C : K'$ and $\Gamma \vdash B =_\beta C : K'$. Finally, by $T\text{-Eq-SYM}$, $T\text{-Eq-TRANS}$, and $S\text{-K-CONV}$, $\Gamma \vdash A =_\beta B : K$. □

4.4 Equality by Subtyping

A consequence of the proof of Anti-Symmetry of the typed operational semantics is that if $\Gamma \vdash_S A \leq B : K$ and $\Gamma \vdash_S B \leq A : K$ then the derivations do not contain any uses of the promotion rule $\text{SWS-Tapp}$. This fact is used to show that:

Lemma 8. If $\Gamma \vdash A =_\beta B : K$ then the derivation of $\Gamma \vdash_S A \leq B : K'$ does not contain any uses of $\text{SWS-Tapp}$, where $\Gamma \vdash_S K \Rightarrow_n K'$.

Proof. By Completeness and by the proof of Anti-Symmetry for the typed operational semantics. □

Furthermore, if one of the subtyping derivations does not use $\text{SWS-Tapp}$ then the other derivation does not either.

Lemma 9.
1. If it is not the case that $\Gamma \vdash_S A \rightarrow_w C : K', C \not\equiv T_\ast$ and $\Gamma \vdash_S B \rightarrow_w T_\ast : K'$ then if the derivation of $\Gamma \vdash_S A \leq B : K'$ contains no uses of $\text{SWS-Tapp}$, then $\Gamma \vdash_S B \leq A : K'$ and the derivation contains no uses of $\text{SWS-Tapp}$.
2. If it is not the case that $A \not\equiv T_\ast$ and $B \equiv T_\ast$ then if the derivation of $\Gamma \vdash_S A \leq_w B : K'$ contains no uses of $\text{SWS-Tapp}$, then $\Gamma \vdash_S B \leq_w A : K'$ and the derivation contains no uses of $\text{SWS-Tapp}$.

Proof. By simultaneous induction on the derivations of $\Gamma \vdash_S A \leq B : K'$ and $\Gamma \vdash_S A \leq_w B : K'$, using that Context Conversion (Lemma 12) creates no new uses of $\text{SWS-Tapp}$ in the cases $\text{SWS-All}$ and $\text{SWS-Tapp}$.

In the light of Lemmas 8 and 9, an algorithm may implement equality and subtyping simultaneously: to check if $A =_\beta B$ it is not necessary to check both $A \leq B$ and $B \leq A$, but is enough to check whether the algorithm uses a step corresponding to promotion (the application rule) in showing $A \leq B$. If it does not then $A =_\beta B$.

The only exception is the case in which the derivation does not contain promotion and is of the form:

$$
\begin{array}{c}
A \rightarrow_w C \\
C \not\equiv T_\ast \\
B \rightarrow_w T_\ast \\
C \leq_w T_\ast
\end{array}
$$

where $C \leq_w T_\ast$ is obtained by the rule $\text{SWS-Top}$. This is the only case in which a subtyping derivation not containing promotion relates two types which are not $\beta$-equal. To exclude this case Lemma 9 Case 1 has the added restrictions on the normal forms of $A$ and $B$. 
5 Replacing Equality with Subtyping

In this section we sketch a presentation of $\mathcal{F}_{\leq}$ without judgemental equality or conversion: we simply add $\beta$-expansion to the left- and right-hand sides of the subtyping judgement, and allow the rules for transitivity and compatible closure of subtyping to handle the extension to full $\beta$-equality. To our knowledge, this is the first such presentation for a higher-order subtyping calculus. This leads to simplifications in the proof of soundness for models of the system, because we remove equality and all of the rules of inference that rely on it.

We write $\Gamma \vdash_{\leq} J$ for judgements derived in the system without an equality judgement, and $\Gamma \vdash_{\leq} A \leq B : K$ if $\Gamma \vdash_{\leq} A \leq B : K$ and $\Gamma \vdash_{\leq} B \leq A : K$.

5.1 Modifications to Inference Rules

The system requires several changes in order to accommodate the removal of equality. First, the inclusion of the equality relation in subtyping, and in particular reflexivity and the rule for $\beta$-equality, needs to be recovered in the rules for subtyping itself:

$$ \Gamma \vdash_{\leq} A : K $$

$$ \Gamma \vdash_{\leq} A \leq A : K \quad (\text{SE-REFL}) $$

$$ \Gamma, X \leq A_1 : K_1 \vdash_{\leq} A_2 : K_2 \quad \Gamma \vdash_{\leq} C \leq A_1 : K_1 $$

$$ \Gamma \vdash_{\leq} A_2[X \leftarrow C] \leq D : K_2[X \leftarrow C] $$

$$ \Gamma \vdash_{\leq} (AX \leq A_1 : K_1.A_2) C \leq D : K_2[X \leftarrow C] \quad (\text{SE-BetaL}) $$

The rule $\text{SE-BetaL}$ similarly introduces a $\beta$-expansion on the right-hand side.

Next, the rule for subtyping applications does not allow the argument to vary. This restriction is for a good reason: allowing subtyping in the argument is unsound. However, we need to recapture the behavior of equality in allowing equal types on the right-hand side of an application:

$$ \Gamma \vdash_{\leq} A \leq B : \Pi X \leq E : K_1.K_2 \quad \Gamma \vdash_{\leq} C \leq D : K_1 $$

$$ \Gamma \vdash_{\leq} C \leq E : K_1 $$

$$ \Gamma \vdash_{\leq} AC \leq BD : K_2[X \leftarrow C] \quad (\text{SE-TApp}) $$

Showing the soundness of this rule for the system with the usual rule $\text{S-TApp}$ will need to use anti-symmetry somewhere, because $\text{S-TApp}$ requires $\beta$-equality of the arguments of the type application.

Finally, we still need an equivalence relation on kinds in order to allow the kinded judgements to vary with respect to equal kinds. We simply lift the equivalence induced by subtyping to kinds. Otherwise, the rules for the new system arise by replacing equality by the new relation $\leq$.

5.2 Changes to Metatheory

We can now show several basic results relating the old presentation with judgemental equality and the new one without it.
First, we can show that equality can be captured by the equivalence relation induced by subtyping.

**Definition 3.** Let \( \Gamma \vdash_S A = B : K \) be defined using the same rules of inference as judgemental equality for \( \Gamma \vdash A = B : K \), but replacing uses of \( \Gamma \vdash A : K \), \( \Gamma \vdash A \leq B : K \), and so on by their corresponding judgement \( \Gamma \vdash_S A : K \), \( \Gamma \vdash_S A \leq B : K \) and so on.

**Lemma 10.** If \( \Gamma \vdash_S A = B : K \) then \( \Gamma \vdash_S A \leq B : K \).

This lemma, plus some minor modifications to the original proof, leads to a proof of Completeness of the new system \( \Gamma \vdash_S J \) for the typed operational semantics.

**Proposition 1 (Completeness).**

1. \( \Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K \) implies \( \Gamma \vdash_S A : K \), \( \Gamma \vdash_S A \equiv_B B : K \) and
   
   \( \Gamma \vdash_S A \equiv_B C : K \).

2. \( \Gamma \vdash_S A \leq B : K \) implies \( \Gamma \vdash_S A, B : K \) and \( \Gamma \vdash_S A \leq B : K \).

We now consider Soundness of \( \Gamma \vdash_S J \) for the typed operational semantics.

**Theorem 2 (Soundness).**

1. If \( \Gamma \vdash_S A : K \) then there are \( K' \), \( B \) and \( C \) such that \( \Gamma \vdash_S K \rightarrow_n K' \) and \( \Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K' \).

2. If \( \Gamma \vdash_S A \equiv_B B : K \) then there are \( C \) and \( K' \) such that \( \Gamma \vdash_S K \rightarrow_n K' \), \( \Gamma \vdash_S A \rightarrow_n C : K' \) and \( \Gamma \vdash_S B \rightarrow_n C : K' \).

3. If \( \Gamma \vdash_S A \leq B : K \) then there is a \( K' \) such that \( \Gamma \vdash_S K \rightarrow_n K' \) and \( \Gamma \vdash_S A \leq B : K' \).

It is probably true that the alternative presentation without equality could be developed by itself, with no reference to equality or anti-symmetry. In our setting, this would require a small change in the rule \texttt{SWS-Ref}, to allow type arguments that are subtypes of each other on the left- and right-hand sides rather than the same normal form. However, the proof of the equivalence of this system with the traditional presentation with equality relies on anti-symmetry.

\section{Conclusions}

We have solved a long-standing open problem, giving a general approach to showing the anti-symmetry of higher-order subtyping. To our knowledge, this is the first proof of anti-symmetry for higher-order subtyping; even for systems like \( F_\omega \), defined roughly ten years ago, the result was unknown. We have also showed this result for the subtyping relation of \( F_\omega \), a higher-order lambda calculus with bounded operator abstraction. Typed operational semantics was essential to our proof, especially the refined understanding of the behavior of types as embodied in the extended rule \texttt{ST-TAR}.
This result has several consequences for the metatheory of type systems with higher-order subtyping. First, it implies that we can now prove the Minimum Types Property, as opposed to the Minimal Types Property which is normally proved. For our system, we can now say what is the relation between all the minimal types of a given term; before we knew that any two minimal types were subtypes of each other, we can now say that they are \( \beta \)-equal. Secondly, we can simplify the basic judgements of type systems with subtyping by eliminating either judgemental equality or conversion.

A practical consequence of this work is that the implementation of higher-order type systems with subtyping can be simplified, because we no longer need to implement either judgemental equality or conversion.

Acknowledgments

We would like to thank Benjamin Pierce and Juliusz Chroboczek for interesting discussions to type equality and anti-symmetry, Giorgio Ghelli for the reference about anti-symmetry for \( F_\leq \), and Jan Zwanenburg for comments on an earlier version. We also thank the anonymous referees for their suggestions.

References

A Results from [15]

Lemma 11 (Adequacy).
1. If \( \Gamma \vdash S K \rightarrow_n K' \) then \( K \rightarrow_\beta K' \).
2. If \( \Gamma \vdash S A \rightarrow_n B \rightarrow_n C : K \) then \( A \rightarrow_\beta B \rightarrow_\beta C \).

Lemma 12 (Context Conversion). If \( \Gamma \vdash S K \rightarrow_n K' \) and \( \Gamma \vdash S A \rightarrow_n A' : K' \) and \( \Gamma, X \leq A : K \vdash S J \) then \( \Gamma, X \leq A' : K' \vdash S J \).

Lemma 13 (Determinacy).
1. If \( \Gamma \vdash S A \rightarrow_n B \rightarrow_n C : K \) and \( \Gamma \vdash S A \rightarrow_n D \rightarrow_n E : K' \) then \( B \equiv D, C \equiv E \) and \( K \equiv K' \).
2. If \( \Gamma \vdash S K \rightarrow_n K' \) and \( \Gamma \vdash S K \rightarrow_n K'' \) then \( K' \equiv K'' \).

Lemma 14 (Subject Reduction). \( \Gamma \vdash S A \rightarrow_n B \rightarrow_n C : K \) and \( A \rightarrow_\beta A' \) imply there is a \( B' \) such that \( B \rightarrow_\beta B' \) and \( \Gamma \vdash S A' \rightarrow_n B' \rightarrow_n C : K \).

Lemma 15 (Subtyping Conversion). Suppose that \( \Gamma \vdash S A \leq W B : K \). Then:
1. If \( \Gamma \vdash S A, A' \rightarrow_n C : K \) then \( \Gamma \vdash S A' \leq W B : K \).
2. If \( \Gamma \vdash S B, B' \rightarrow_n C : K \) then \( \Gamma \vdash S A \leq W B' : K \).

Similarly for \( \Gamma \vdash S A \leq B : K \).

Proposition 2 (Completeness).
1. \( \Gamma \vdash S K \rightarrow_n K' \) implies \( \Gamma \vdash K \) and \( \Gamma \vdash K \equiv_\beta K' \).
2. \( \Gamma \vdash S A \rightarrow_n B \rightarrow_n C : K \) implies \( \Gamma \vdash A : K, \Gamma \vdash A \equiv_\beta B : K \) and \( \Gamma \vdash A \equiv_\beta C : K \).
3. \( \Gamma \vdash S A \leq W B : K \) implies \( \Gamma \vdash A \leq B : K \).
4. \( \Gamma \vdash S A \leq B : K \) implies \( \Gamma \vdash A, B : K \) and \( \Gamma \vdash A \leq B : K \).

Theorem 3 (Soundness).
1. If \( \Gamma \vdash K \) then there is a \( K' \) such that \( \Gamma \vdash S K \rightarrow_n K' \); if \( \Gamma \vdash K \equiv_\beta K' \) then there is a \( K'' \) such that \( \Gamma \vdash S K \rightarrow_n K'' \) and \( \Gamma \vdash S K' \rightarrow_n K'' \).
2. If \( \Gamma \vdash A : K \) then there are \( K', B \) and \( C \) such that \( \Gamma \vdash S K \rightarrow_n K' \) and \( \Gamma \vdash S A \rightarrow_n B \rightarrow_n C : K' \); if \( \Gamma \vdash A \equiv_\beta B : K \) then there are \( C, K' \) and \( \Gamma \vdash S K \rightarrow_n K' \) and \( \Gamma \vdash S A \rightarrow_n C : K' \) and \( \Gamma \vdash S B \rightarrow_n C : K' \).
3. If \( \Gamma \vdash A \leq B : K \) then there is a \( K' \) such that \( \Gamma \vdash S K \rightarrow_n K' \) and \( \Gamma \vdash S A \leq B : K' \).

Lemma 16 (Upper Bound).
If \( \Gamma \vdash X(A_1, \ldots, A_m) : K \) then \( \Gamma \vdash \Gamma(X)(A_1, \ldots, A_m) : K \).

Lemma 17. If \( \Gamma \vdash S A \leq B : K \) then \( \Gamma \vdash S A : K \) and \( \Gamma \vdash S B : K \).
B  Rules for $\mathcal{F}_\leq$

Equality Rules

The equality rules for $\mathcal{F}_\leq$ include typed rules of inference stating that $\Gamma \vdash A \equiv B : K$ is an equivalence relation and a compatible closure for $A \rightarrow B$, $\forall X \leq A_1 : K.A_2$, $\lambda X \leq A_1 : K.A_2$, and type application, in addition to the following two rules:

$$
\frac{\Gamma, X \leq A_1 : K_1 \vdash A_2 : K_2 \quad \Gamma \vdash C \leq A_1 : K_1}{\Gamma \vdash (\lambda X \leq A_1 : K_1.A_2)[X\leftarrow C] : K_2 \leq K \leq K_1} \quad (T\text{-Eq} \cdot \text{Beta})
$$

$$
\frac{\Gamma \vdash A \equiv B : K \quad \Gamma \vdash K \equiv K' : K}{\Gamma \vdash A \equiv B : K'} \quad (T\text{-Eq} \cdot \text{Conv})
$$

Subtyping Rules

The subtyping rules for $\mathcal{F}_\leq$ include typed rules stating that $\Gamma \vdash A \leq B : K$ is transitive, that it includes the relation $\Gamma \vdash A \equiv B : K$, that $T_K$ is greater than all $A$ of kind $K$, and the following rules:

$$
\frac{\Gamma_1, X \leq A : K, \Gamma_2 \vdash \text{ok}}{\Gamma_1, X \leq A : K, \Gamma_2 \vdash X \leq A : K} \quad (S\text{-Var})
$$

$$
\frac{\Gamma \vdash B_1 \leq A_1 : \ast \quad \Gamma \vdash A_2 \leq B_2 : \ast}{\Gamma \vdash A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2 : \ast} \quad (S\text{-Arrow})
$$

$$
\frac{\Gamma, X \leq C : K \vdash A \leq B : \ast}{\Gamma, X \leq C : K \vdash \forall X \leq C : K.A \leq \forall X \leq C : K.B : \ast} \quad (S\text{-All})
$$

$$
\frac{\Gamma \vdash A \leq B : \Pi X \leq C : K_1.A \leq \Pi X \leq C : K_1.B : \Pi X \leq C : K_1.K_2}{\Gamma \vdash A \leq \Pi X \leq D : K_1.K_2 \quad \Gamma \vdash C \leq D : K_1}{\Gamma \vdash AC \leq BC : K_2[X\leftarrow C]} \quad (S\text{-App})
$$

$$
\frac{\Gamma \vdash A \leq B : K \quad \Gamma \vdash K \equiv K' : K'}{\Gamma \vdash A \leq B : K'} \quad (S\text{-Conv})
$$

C  Rules for the Typed Operational Semantics

Type Reduction Rules

$$
\frac{}{\Gamma \vdash_s \text{ok}} \quad (ST\cdot\text{Top})
$$

$$
\frac{\Gamma \vdash_s X \rightarrow_n X'}{\Gamma \vdash_s X \rightarrow_n X : K'} \quad (ST\cdot\text{Var})
$$

$$
\frac{\Gamma \vdash_s A : K' \quad \Gamma \vdash_s K \rightarrow_n K' \quad (X \leq A : K) \in \Gamma}{\Gamma \vdash_s X \rightarrow_n X : K'} \quad (ST\cdot\text{Van})
$$

$$
\frac{\Gamma \vdash_s A \rightarrow_n X(A_1, \ldots, A_n) : \Pi Y \leq C : K_1.K_2 \quad \Gamma \vdash_s \Gamma(X) \rightarrow_n D : K'}{\Gamma \vdash_s D(A_1, \ldots, A_n, F) : K} \quad (ST\cdot\text{App})
$$

$$
\frac{\Gamma \vdash_s E \leq_n C : K_1 \quad \Gamma \vdash_s K_2[Y\leftarrow B] \rightarrow_n K}{\Gamma \vdash_s AB \rightarrow_n X(A_1, \ldots, A_n, F) \rightarrow_n X(A_1, \ldots, A_n, F) : K} \quad (ST\cdot\text{TApp})
$$
\[
\begin{array}{ll}
\Gamma \vdash_S A_1 \rightarrow_n B_1 : \ast & \Gamma \vdash_S A_2 \rightarrow_n B_2 : \ast \\
\Gamma \vdash_S (A_1 \rightarrow A_2) \rightarrow_n (A_1 \rightarrow A_2) \rightarrow_n (B_1 \rightarrow B_2) : \ast \\
\Gamma \vdash_S A \rightarrow_n C : K' & \Gamma \vdash_S K \rightarrow_n K' \\
\Gamma, X \leq A; K \vdash_S B \rightarrow_n D : \ast \\
\Gamma \vdash_S \forall X \leq A; K, B \rightarrow_n \forall X \leq A; K, B \rightarrow_n \forall X \leq C; K', D : \ast \\
\Gamma \vdash_S A \rightarrow_n C : K_1' & \Gamma \vdash_S K_1 \rightarrow_n K_1' \\
\Gamma, X \leq A; K_1 \vdash_S B \rightarrow_n D : K_2 \\
\Gamma \vdash_S \Lambda X \leq A; K, B \rightarrow_n \Lambda X \leq A; K, B \rightarrow_n \Lambda X \leq C; K', D : \Pi X \leq C; K_1', K_2 \\
\Gamma \vdash_S B \rightarrow_n \Lambda X \leq A; K_1, D : \Pi X \leq A'; K_1', K_2 & \Gamma \vdash_S K_2[X \leftarrow C] \rightarrow_n K \\
\Gamma \vdash_S D[X \leftarrow C] \rightarrow_n E \rightarrow_n F : K & \Gamma \vdash_S C \leq A : K_1' \\
\Gamma \vdash_S B \rightarrow_n C \rightarrow_n E \rightarrow_n F : K \\
\end{array}
\]

**Weak-Head Subtyping and Subtyping**

\[
\begin{array}{ll}
\Gamma \vdash_S A \rightarrow_n B : \ast & \text{HV}(A) \text{ undefined} \\
\Gamma \vdash_S A \leq W \rightarrow_n T : \ast \\
\Gamma \vdash_S X(A_1, \ldots, A_m) \rightarrow_n C : K & \Gamma \vdash_S \Gamma(X) \rightarrow_n B : K' \\
\Gamma \vdash_S B(A_1, \ldots, A_m) \rightarrow_n E : K & \Gamma \vdash_S E \leq_W A : K \\
A \neq X(A_1, \ldots, A_m) \\
\Gamma \vdash_S X(A_1, \ldots, A_m) \leq_W A : K \\
\Gamma \vdash_S X(A_1, \ldots, A_m) \rightarrow_n B : K \\
\Gamma \vdash_S \forall X \leq A_1; \ldots, A_m \rightarrow_n B : K \\
\Gamma \vdash_S B_1 \leq A_1 : \ast & \Gamma \vdash_S A_2 \leq B_2 : \ast \\
\Gamma \vdash_S \forall X \leq A_1; \ldots, A_m \rightarrow_n B_1 \rightarrow_B : \ast \\
\Gamma, X \leq A_1; K \vdash_S A_2 \leq B_2 : \ast & \Gamma \vdash_S K, K' \rightarrow_n K'' \\
\Gamma \vdash_S A_1, B_1 \rightarrow_n C : K'' \\
\Gamma \vdash_S \forall X \leq A_1; K, A_2 \leq W \rightarrow_n \forall X \leq B_1; K' \rightarrow_B : \ast \\
\Gamma, X \leq A_1; K_1 \vdash_S A_2 \leq B_2 ; K_2 & \Gamma \vdash_S K_1, K_1' \rightarrow_n K'' \\
\Gamma \vdash_S A_1, B_1 \rightarrow_n C : K'' \\
\Gamma \vdash_S \forall X \leq A_1; K, A_2 \leq W \rightarrow_n \forall X \leq B_1; K_1', B_2 ; \Pi X \leq C; K', K_2 \\
\Gamma \vdash_S A \rightarrow_n C : K & \Gamma \vdash_S B \rightarrow_n D : K & \Gamma \vdash_S C \leq W D : K \\
\Gamma \vdash_S A \leq B : K \\
\end{array}
\]