Relating Church-Style and Curry-Style Subtyping

Adriana Compagnoni
Stevens Institute of Technology
Castle Point on Hudson
Hoboken NJ 07030 USA
abc@cs.stevens.edu

Healfdene Goguen
Google Inc.
111 8th Ave.
New York, NY 10011 USA
hhg@google.com

Type theories with higher-order subtyping or singleton types are examples of systems where computation rules for variables are affected by type information in the context. A complication for these systems is that bounds declared in the context do not interact well with the logical relation proof of completeness or termination. This paper proposes a natural modification to the type syntax for $F^{\omega}_{\leq}$, adding a variable's bound to the variable type constructor, thereby separating the computational behavior of the variable from the context. The algorithm for subtyping in $F^{\omega}_{\leq}$ can then be given on types without context or kind information. As a consequence, the metatheory follows the general approach for type systems without computational information in the context, including a simple logical relation definition without Kripke-style indexing by context. This new presentation of the system is shown to be equivalent to the traditional presentation without bounds on the variable type constructor.

1 Introduction

Logical relations are a powerful technique for proving metatheoretic properties of type theories. The traditional approach to the metatheory of type theories, for example that of Pure Type Systems [2], studies properties of untyped reduction and conversion, and then completes the study of type-checking by proving strong normalization with a logical relation construction.

This approach has been difficult to adapt to systems where variables may behave differently according to context information. Examples where this may occur are type systems with singleton types, where a variable of a singleton type is equal to the unique element of the type, or subtyping, where a type variable may be replaced by its bound in a derivation of subtyping. The key difficulty is that strong normalization of a term or termination of subtyping on higher-order types depends on information in the context, but that normalization or termination also needs to be closed under replacement by equal contexts, in order to model the constructors that introduce the computational information into the context.

For example, in $F^{\omega}_{\leq}$, consider a putative proof of strong normalization in the case of a derivation of $X \leq A : \star \to \star \vdash X(C) : \star$. Such a proof would have a hypothesis that $A(C)$ is strongly normalizing, since the model must allow a variable to be replaced by its bound, referred to as promotion. However, to model the rule that $\vdash \forall X \leq A : \star \to \star . X(C) = \forall X \leq B : \star \to \star . X(C) : \star$, we would need that $B(C)$ is strongly normalizing for arbitrary B such that $\vdash A = B : \star \to \star$ *before* constructing the model; the behavior of $X(C)$ varies according to its context.

Several papers have addressed systems of this type, but each of these approaches differs from the usual approach to metatheory of type theories. Compagnoni and Goguen [7, 6] use an algorithm where a variable's bound is normalized before promoting the variable. This allows context replacement to be proved before the completeness proof, but it seems to be an odd requirement and was only introduced to get the proof to work. Furthermore, the algorithm is less efficient than an algorithm that postpones normalization of the bound. Stone and Harper [13] prove termination for an algorithm for singletons using the unnormalized singleton rather than normalizing it first. Their Kripke-style proof indexes the

E. Pimentel, B. Venneri and J. Wells (Eds.): Workshop on
Intersection Types and Related Systems 2010 (ITRS 2010).
EPTCS 45, 2011, pp. 1–15, doi:10.4204/EPTCS.45.1

model with sets of possible contexts in which a term is well-typed. In the example above, the possible B are limited by considering contexts that arise from bounds introduced by the $\forall$ constructor. This differs from the standard Kripke-style proof of strong normalization, which is relative to a single context.

In this paper we propose separating the computational behavior of variables from the context. We introduce a modified type structure for $F^{\omega}_{\leq}$ [3, 4, 11], where the type constructor for variables is X_A with the variable's bound A explicitly mentioned. We call this presentation "à la Church" for its obvious similarity to type labels on λ-abstractions, and we call the traditional presentation "à la Curry". With this change, the term structure tells us how promotion will be used without reference to the context.

This presentation allows us to give a kind- and context-free definition of the algorithm for subtyping, since the only use of the context in the traditional algorithm is when a variable is replaced with its bound. This in turn leads to an approach to the metatheory consistent with the usual approach for type theories, since promoting a variable to a type convertible with its bound, the cause of the difficulties in the system without bounded variables, is never necessary. In our example above, the terms would be $\forall X \leq A : \star \to \star . X_A(C) = \forall X \leq B : \star \to \star . X_B(C)$: the behavior of $X_A(C)$ and $X_B(C)$ is fixed regardless of context.

While changing the term structure could be considered a syntactic trick, in our opinion our presentation points to a deficiency in the syntax of the traditional term structure of $F^{\omega}_{\leq}$. In general, model constructions work best when there is a close relationship between terms and derivations: this is best illustrated by Streicher's extended term structure and partial interpretation for the Calculus of Constructions [14]. The inability to construct traditional models to show decidability of higher-order subtyping for the expected algorithm suggests that the type structure of $F^{\omega}_{\leq}$ is inappropriate. We believe that the trick is that the syntax without bounded type variables works at all. The equivalence of the two presentations shows that the additional information necessary for the model construction can be ignored in programs.

The system Full $F^{\omega}_{\leq}$, with contravariance in the bounds of quantified types, further illustrates our point. Surprisingly, in contrast to the system with unlabeled type variables, the algorithm for subtyping for Full $F^{\omega}_{\leq}$ with bounded variables cannot be defined. Essentially, when the implicit contravariant type information in the unlabeled types is made explicit, the side conditions distinguishing variable reflexivity from promotion cannot be expressed in a valid inductive definition. We shall discuss the technical reasons that the definition fails in more detail when we define the algorithm for Kernel $F^{\omega}_{\leq}$. However, the inability to define the algorithm for Full $F^{\omega}_{\leq}$ over the explicit type structure gives a strong indication that the unlabeled type structure is an inadequate representation of types for $F^{\omega}_{\leq}$, rather than the addition of bound information to the type being a trick.

The correctness of the traditional system à la Curry is a consequence of the equivalence of the two presentations. There are two substantial differences in the treatment of the new system. First, because variables mention their bound explicitly, the new presentation exposes the difference between the operations of renaming $[X \leftarrow Y]$, which changes variable names but does not change the bound, and substitution of a variable, $[Y_B/X]$, which replaces the bound of X. Secondly, the subtyping judgement is needed in the formulation of the inference rules for the kinding judgement, for example in the rule TVAR for kinding a type variable: this is not necessary in the traditional presentation.

In this paper we address all of the complications listed above. We first study the properties of the subtyping relation, including completeness and correctness, anti-symmetry, transitivity elimination and decidability. We complete our development by showing that our system is equivalent to the traditional one without bounds. We ignore completely the term language, since its metatheory is standard once decidability of subtyping has been proved. As such, we do not treat substitution for bounded variables as occur in $\forall$, since this substitution only occurs in the reduction relation for terms.

2 Syntax

We now present the term constructors, judgements and rules of inference for kinding and subtyping in $F^{\omega}_{\leq}$.

2.1 Syntactic Categories

The kinds of $F^{\omega}_{\leq}$ are the kind $\star$ of proper types and the kinds $K \to K'$ of functions on types and type operators. We assume an infinite collection of type variable names $X, Y, Z, \ldots$. The types include variables with explicit bounds X_A; the top type $\mathrm{T}_\star$; function types $A \to B$; and types $\forall X \leq A : K.B$ of polymorphic functions, in which the bound type variable X ranges over all subtypes of the upper bound A. Moreover, like F^{ω}, we allow types to be abstracted on types, of the form $\Lambda X : K.A$, and we can apply types to argument types $A(B)$. Contexts Γ, Δ are either the empty context $()$ or extended contexts $\Gamma, X \leq A : K$.

We identify types that differ only in the names of bound variables. We write $A(B_1, \ldots, B_n)$ for $(A(B_1))\ldots(B_n)$. If $A \equiv X_C(B_1, \ldots, B_n)$ then A has head variable X_C; we write $\mathrm{HV}(-)$ for the partial function returning the head variable of a type. We also extend the top type $\mathrm{T}_\star$ to any kind K by defining inductively $\mathrm{T}_{K \to K'} = \Lambda X : K.\mathrm{T}_{K'}$. We use $X : K$ as an abbreviation for $X \leq \mathrm{T}_K : K$ in contexts; in this case we say X is a variable without a bound.

Because type variables are decorated with their bounds, we need to be careful with our definition of substitution: specifically, a renaming should be restricted to renaming the variables in the bound A of a variable X_A, as opposed to changing the bound as may occur in a substitution of Y_B for X in X_A. We therefore define parallel substitutions γ, δ as either the empty substitution $()$; the extension of a parallel substitution γ with a renaming of a variable X by another variable Y, written $\gamma[X \leftarrow Y]$; or the extension of a parallel substitution γ with a substitution of a variable X by a type A, written $\gamma[A/X]$. We say γ is a renaming if $\gamma = ()$ or if $\gamma = \gamma_0[X \leftarrow Y]$ with γ_0 a renaming. We write id_Γ for the identity renaming of the type variables declared in Γ.

We write $B[\gamma]$ for the capture-avoiding simultaneous replacement of each of the variables by its corresponding value, defined as follows on variables and lifted in the usual way to arbitrary types:

- $X_A[()] = X_A$.
- $X_A[\gamma[X \leftarrow Y]] = Y_{A[\gamma[X \leftarrow Y]]}$.
- $X_A[\gamma[Y \leftarrow Z]] = X_{A[\gamma[Y \leftarrow Z]]}$, if $X \neq Y$.
- $X_A[\gamma[B/X]] = B$.
- $X_A[\gamma[B/Y]] = X_{A[\gamma[B/Y]]}$, if $X \neq Y$.

Observe that B cannot be a variable Y or Z in the last two equations, but must instead be a bounded type variable Y_C or Z_D.

We also write $B[A/X]$ for the parallel substitution that is the identity renaming on the free variables in B other than X, and the substitution of X by A. We have standard properties of parallel substitution, for example that $A[\gamma][\delta] \equiv A[\gamma \circ \delta]$ and $(\gamma[A/X]) \circ \delta = (\gamma \circ \delta)[A\delta/X]$. We also write $A \rhd B$ for the standard notion of one-step β-reduction. We have the standard property of Church–Rosser for reduction.

2.2 Judgements and Rules of Inference

The judgement forms are $\Gamma \vdash A : K$ for well-kinded types and $\Gamma \vdash A \leq B : K$ for subtyping. We sometimes write $\Gamma \vdash \mathrm{ok}$ for $\Gamma \vdash \mathrm{T}_\star : \star$, formalizing the well-formedness of Γ, and $\Gamma \vdash A = B : K$ for $\Gamma \vdash A \leq B : K$

and $\Gamma \vdash B \leq A : K$. We may also use the metavariable J to range over statements (right-hand sides of judgements) of any of these judgement forms.

The rules of inference are presented as simultaneously defined inductive relations over the judgements. We start with several admissible structural rules, and follow with the kinding and subtyping rules.

2.2.1 Kinding Rules

The following rules formalize the judgement $\Gamma \vdash A : K$, stating that the type A is well-formed and of kind K in context Γ.

$$() \vdash \mathrm{T}_\star : \star \qquad \text{(TopEmp)}$$

$$\frac{\Gamma \vdash A : K \qquad X \notin \mathrm{dom}(\Gamma)}{\Gamma, X \leq A : K \vdash \mathrm{T}_\star : \star} \qquad \text{(TopExt)}$$

$$\frac{\Gamma \vdash B : K \qquad \Gamma \vdash A \leq B : K \qquad X \leq A : K \in \Gamma}{\Gamma \vdash X_B : K} \qquad \text{(TVar)}$$

$$\frac{\Gamma, X : K \vdash A : K'}{\Gamma \vdash \Lambda X : K.A : K \to K'} \qquad \text{(TAbs)}$$

$$\frac{\Gamma \vdash A : K \to K' \qquad \Gamma \vdash B : K}{\Gamma \vdash A(B) : K'} \qquad \text{(TApp)}$$

$$\frac{\Gamma \vdash A : \star \qquad \Gamma \vdash B : \star}{\Gamma \vdash A \to B : \star} \qquad \text{(Arrow)}$$

$$\frac{\Gamma, X \leq A : K \vdash B : \star}{\Gamma \vdash \forall X \leq A : K.B : \star} \qquad \text{(All)}$$

Notice that in rule TVar it is possible that $X \in \mathrm{FV}(B)$ when $\Gamma \vdash X_B : K$. The premise that $\Gamma \vdash B : K$ in this rule ensures that Subject Reduction goes through smoothly, without needing to refer to subtyping.

2.2.2 Subtyping Rules

Finally, the following rules formalize the judgement $\Gamma \vdash A \leq B : K$, stating that type A is a subtype of type B and both are well-formed of kind K in context Γ.

$$\frac{\Gamma \vdash A : K}{\Gamma \vdash A \leq A : K} \qquad \text{(S-Refl)}$$

$$\frac{\Gamma \vdash A \leq B : K \qquad \Gamma \vdash B \leq C : K}{\Gamma \vdash A \leq C : K} \qquad \text{(S-Trans)}$$

$$\frac{\Gamma \vdash A : K}{\Gamma \vdash A \leq \mathrm{T}_K : K} \qquad \text{(S-Top)}$$

$$\frac{\Gamma \vdash A \leq B : K \qquad \Gamma \vdash B = C : K \qquad X \leq A : K \in \Gamma}{\Gamma \vdash X_B \leq X_C : K} \qquad \text{(S-TVar)}$$

$$\frac{\Gamma \vdash A \leq B : K \qquad \Gamma \vdash B : K \qquad X \leq A : K \in \Gamma}{\Gamma \vdash X_B \leq B : K} \qquad \text{(S-Promote)}$$

$$\frac{\Gamma, X : K \vdash A \leq B : K'}{\Gamma \vdash \Lambda X : K.A \leq \Lambda X : K.B : K \to K'} \qquad \text{(S-TAbs)}$$

$$\frac{\Gamma \vdash A \leq C : K \to K' \qquad \Gamma \vdash B = D : K}{\Gamma \vdash A(B) \leq C(D) : K'} \tag{S-TApp}$$

$$\frac{\Gamma \vdash B_1 \leq A_1 : \star \qquad \Gamma \vdash A_2 \leq B_2 : \star}{\Gamma \vdash A_1 \to A_2 \leq B_1 \to B_2 : \star} \tag{S-Arrow}$$

$$\frac{\Gamma \vdash A = C : K \qquad \Gamma, X \leq A : K \vdash B \leq D : \star}{\Gamma \vdash \forall X \leq A : K.B \leq \forall X \leq C : K.D : \star} \tag{S-All}$$

$$\frac{\Gamma, X : K \vdash A : K' \qquad \Gamma \vdash B : K}{\Gamma \vdash (\Lambda X : K.A)(B) \leq A[B/X] : K'} \tag{S-BetaL}$$

$$\frac{\Gamma, X : K \vdash A : K' \qquad \Gamma \vdash B : K}{\Gamma \vdash A[B/X] \leq (\Lambda X : K.A)(B) : K'} \tag{S-BetaR}$$

3 The Algorithm

In this section we define the algorithm for kinding and subtyping.

First, we define the relations $\to_w$ for weak-head reduction and $\twoheadrightarrow_n$ for reduction to normal form.

Definition 3.1

- $(\Lambda X : K.A)(B) \to_w A[B/X]$.
- $A(B) \to_w C(B)$ *if* $A \to_w C$.
- $\mathrm{T}_\star \twoheadrightarrow_n \mathrm{T}_\star$.
- $X_A(B_1, \ldots, B_n) \twoheadrightarrow_n X_C(D_1, \ldots, D_n)$ *if* $A \twoheadrightarrow_n C$ *and* $B_i \twoheadrightarrow_n D_i$ *for* $1 \leq i \leq n$.
- $\Lambda X : K.A \twoheadrightarrow_n \Lambda X : K.B$ *if* $A \twoheadrightarrow_n B$.
- $A \to B \twoheadrightarrow_n C \to D$ *if* $A \twoheadrightarrow_n C$ *and* $B \twoheadrightarrow_n D$.
- $\forall X \leq A : K.B \twoheadrightarrow_n \forall X \leq C : K.D$ *if* $A \twoheadrightarrow_n C$ *and* $B \twoheadrightarrow_n D$.
- $A \twoheadrightarrow_n C$ *if* $A \to_w B$ *and* $B \twoheadrightarrow_n C$.

We also write $A \downarrow_n B$ *iff there is a* C *such that* $A \twoheadrightarrow_n C$ *and* $B \twoheadrightarrow_n C$, *and* $A \downarrow_n$ *iff there is a* C *such that* $A \twoheadrightarrow_n C$.

Lemma 3.2 *If* $A \twoheadrightarrow_n C$ *and* $A \rhd B$ *then* $B \twoheadrightarrow_n C$.

The algorithm has a judgement for kinding, $\Gamma \vdash_A A : K$, and two judgements for subtyping, $\vdash_A A \leq_W B$ for subtyping weak-head normal forms, and $\vdash_A A \leq B$ for subtyping arbitrary types. The judgement $\Gamma \vdash_A A : K$ corresponds to type inference: the context Γ and type A are inputs, and the kind K is an output. The algorithm for subtyping is analogous to untyped conversion in the λ-calculus: it is purely a computational relation, without reference to kind information. Furthermore, the algorithm for kinding does not refer to subtyping, because subtyping is used for the term language of $F^{\omega}_{\leq}$ and not for types.

The algorithm is defined by the following rules of inference. It is syntax-directed, and it will be shown to be terminating on well-formed types. Clearly types do not need to be well-formed to be subjects of the algorithmic subtyping judgement. Since algorithmic subtyping incorporates weak-head reduction, it is also clearly not terminating in general.

$$\Gamma \vdash_A \mathrm{T}_\star : \star \tag{AT-Top}$$

$$\frac{X \leq A : K \in \Gamma}{\Gamma \vdash_A X_A : K} \tag{AT-TVar}$$

$$\frac{\Gamma, X : K \vdash_A A : K' \qquad X \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash_A \Lambda X : K.A : K \to K'} \tag{AT-TAbs}$$

$$\frac{\Gamma \vdash_A A : K \to K' \qquad \Gamma \vdash_A B : K}{\Gamma \vdash_A A(B) : K'} \tag{AT-TApp}$$

$$\frac{\Gamma \vdash_A A : \star \qquad \Gamma \vdash_A B : \star}{\Gamma \vdash_A A \to B : \star} \tag{AT-Arrow}$$

$$\frac{\Gamma \vdash_A A_1 : K \quad \Gamma, X \leq A_1 : K \vdash_A A_2 : \star \qquad X \notin \mathrm{dom}(\Gamma)}{\Gamma \vdash_A \forall X \leq A_1 : K.A_2 : \star} \tag{AT-All}$$

$$\frac{\mathrm{HV}(A) \text{ undefined and } A \text{ is not an abstraction}}{\vdash_A A \leq_W \mathrm{T}_\star} \tag{AWS-Top}$$

$$\frac{X_A(B_1,...,B_n) \downarrow_n X_C(D_1,...,D_n)}{\vdash_A X_A(B_1,...,B_n) \leq_W X_C(D_1,...,D_n)} \tag{AWS-TVar}$$

$$\frac{\vdash_A A(B_1,...,B_n) \leq C \qquad C \not\downarrow_n X_A(B_1,...,B_n)}{\vdash_A X_A(B_1,...,B_n) \leq_W C} \tag{AWS-Promote}$$

$$\frac{\vdash_A A \leq B}{\vdash_A \Lambda X : K.A \leq_W \Lambda X : K.B} \tag{AWS-TAbs}$$

$$\frac{\vdash_A B_1 \leq A_1 \qquad \vdash_A A_2 \leq B_2}{\vdash_A A_1 \to A_2 \leq_W B_1 \to B_2} \tag{AWS-Arrow}$$

$$\frac{A_1 \downarrow_n B_1 \qquad \vdash_A A_2 \leq B_2}{\vdash_A \forall X \leq A_1 : K.A_2 \leq_W \forall X \leq B_1 : K.B_2} \tag{AWS-All}$$

$$\frac{A \twoheadrightarrow_w C \qquad B \twoheadrightarrow_w D \qquad \vdash_A C \leq_W D}{\vdash_A A \leq B} \tag{AS-Inc}$$

The rule for subtyping of bounded variables, S-TVar, states that X_A is less than X_B if A and B are equal. The algorithmic presentation of this system needs a side condition, independent of the algorithm, to determine whether to apply rule AWS-TVar or rule AWS-Promote: in our presentation of Kernel $F^{\omega}_{\leq}$, this is whether the left- and right-hand sides are convertible. On the other hand, a translation of Full $F^{\omega}_{\leq}$ with bounded variables requires the premise $\Gamma \vdash B \leq A : K$ in AWS-TVar to demonstrate the equivalence of the explicit and unlabeled type presentations. However, the algorithm cannot be defined with this premise, because the negation of this premise in AWS-Promote requires a negative occurrence of the judgement being defined, which is not a valid inductive definition.

Also, observe that while the comparison $A \not\downarrow_n X_A(B_1,...,B_n)$ in AWS-Promote might hold if either side diverges, in practice the algorithm will always be applied to well-formed terms, which will be shown to be terminating.

4 Metatheory

In this section we develop the basic metatheory for the algorithm.

We begin with the relations $A >_P B$, formalizing a use of promotion, and $A >_S B$, formalizing B an immediate subterm of A, both for A weak-head normal.

Definition 4.1

- $X_A(B_1,...,B_n) >_P A(B_1,...,B_n)$.
- – $\Lambda X : K.A >_S A$.

– $A_1 \to A_2 >_S A_1$ *and* $A_1 \to A_2 >_S A_2$.
– $\forall X \leq A_1 : K.A_2 >_S A_2$.

Definition 4.2 (Strong Normalization, Termination) *We define the following predicates inductively:*

- $\mathrm{SN}(A)$ *iff* $\mathrm{SN}(B)$ *for all* B *such that* $A \rhd B$.
- $T(A)$ *iff* $T(B)$ *for all* B *such that* $A \rhd B$, $A >_P B$ *or* $A >_S B$.

The predicate $T(A)$, or A is terminating, formalizes the possible types that the algorithm may encounter when invoked on a judgement containing A. As for strong normalization, a base case for $T(A)$ would be a type with no reducts.

Lemma 4.3 *We have the following properties of* $T(-)$ *and* $\mathrm{SN}(-)$*:*

1. $T(A)$ *implies* $\mathrm{SN}(A)$.
2. $T(A)$ *implies* $A \downarrow_n$.
3. *If* $T(A)$ *and* $A \rhd B$ *then* $T(B)$.
4. $T(\mathrm{T}_\star)$.
5. $T(A)$ *iff* $T(\Lambda X : K.A)$.
6. $T(\mathrm{T}_K)$.
7. $T(A)$ *and* $T(B)$ *iff* $T(A \to B)$.
8. $\mathrm{SN}(A)$ *and* $T(B)$ *iff* $T(\forall X \leq A : K.B)$.
9. *If* $A >_P B$ *then* $T(A)$ *iff* $T(B)$.
10. $\mathrm{SN}(A_i)$ *for* $1 \leq i \leq n$ *iff* $T(X_{\mathrm{T}_K}(A_1, ..., A_n))$.
11. *If* $T(A)$, $T(B)$, $A(B) \to_w C$ *and* $T(C)$ *then* $T(A(B))$.

Proof The only case that is difficult is Case 11, which follows by standard λ-calculus properties.

Proposition 4.4 (Decidability) *If* $T(A)$ *and* $T(B)$ *then* $\vdash_A A \leq B$ *and* $\vdash_A A \leq_W B$ *terminate.*

Proof By induction on the sum of the length of the derivations of $T(A)$ and $T(B)$.

There are two cases: either A or B has a weak-head reduct or they are both weak-head normal. In the first case the result follows by induction hypothesis. Otherwise, by inspection:

- $A \equiv \mathrm{T}_\star$. If $B \equiv \mathrm{T}_\star$ then $\vdash_A A \leq B$ succeeds, otherwise it fails.
- $A \equiv X_C(D_1, ..., D_n)$. If $B \equiv X_E(F_1, ..., F_n)$ then $C \downarrow_n E$ and $D_i \downarrow_n F_i$ terminate, since $T(A)$ implies $A \downarrow_n$ and $T(B)$ implies $B \downarrow_n$, so if these conditions hold then $\vdash_A A \leq B$ succeeds.

  Otherwise, $X_C(D_1, ..., D_n) >_P C(D_1, ..., D_n)$, so $\vdash_A C(D_1, ..., D_n) \leq B$ terminates by induction hypothesis, so $\vdash_A A \leq B$ succeeds or fails as $\vdash_A C(D_1, ..., D_n) \leq B$ does.
- $A \equiv A_1 \to A_2$. If $B \equiv \mathrm{T}_\star$ then $\vdash_A A \leq B$ succeeds. If $B \equiv B_1 \to B_2$ then $\vdash_A B_1 \leq A_1$ and $\vdash_A A_2 \leq B_2$ terminate by induction hypothesis, since $A_1 \to A_2 >_S A_i$ for $i \in \{1, 2\}$ and similarly for B, so $\vdash_A A_1 \to A_2 \leq B_1 \to B_2$ terminates.

  Otherwise, $\vdash_A A \leq B$ fails.

- $A \equiv \forall X \leq A_1 : K.A_2$. If $B \equiv \mathrm{T}_\star$ then $\vdash_A A \leq \mathrm{T}_\star$ succeeds. If $B \equiv \forall X \leq B_1 : K.B_2$ then $A_1 \downarrow_n B_1$ terminates because $T(A_1)$ and $T(B_1)$ imply $A_1 \downarrow_n$ and $B_1 \downarrow_n$. Furthermore, $\forall X \leq A_1 : K.A_2 >_S A_2$ and $\forall X \leq B_1 : K.B_2 >_S B_2$, so $\vdash_A A_2 \leq B_2$ terminates by induction hypothesis, and $\vdash_A \forall X \leq A_1 : K.A_2 \leq \forall X \leq B_1 : K.B_2$ succeeds or fails as $\vdash_A A_2 \leq B_2$ does.

  Otherwise, $\vdash_A A \leq B$ fails.
- $A \equiv \Lambda X : K.A_1$. If $B \equiv \Lambda X : K.B_1$ then $\Lambda X : K.A_0 >_S A_0$ and $\Lambda X : K.B_0 >_S B_0$, so $\vdash_A A_1 \leq B_1$ terminates by induction hypothesis, and $\vdash_A \forall X : K.A_0 \leq \forall X : K.B_0$ succeeds or fails as $\vdash_A A_1 \leq B_1$ does.

  Otherwise, $\vdash_A A \leq B$ fails.

Lemma 4.5 (Reflexivity) *If $T(A)$ then $\vdash_A A \leq A$.*

Proof We show $\vdash_A A \leq_W A$ for weak-head normal A and $\vdash_A A \leq A$ for all A, by induction on $T(A)$.

If A is weak-head normal then the proof proceeds by case analysis. For example, suppose $A \equiv X_B(C_1,...,C_n)$. Then $\mathrm{SN}(B)$ and $\mathrm{SN}(C_i)$ for $1 \leq i \leq n$, so $B \downarrow_n B$ and $C_i \downarrow_n C_i$, so $\vdash_A X_B(C_1,...,C_n) \leq_W X_B(C_1,...,C_n)$.

For arbitrary A, $A \twoheadrightarrow_w B$ and $\vdash_A B \leq_W B$ immediately if A is weak-head normal or otherwise by induction hypothesis.

Lemma 4.6 (Subject Conversion) *If $\vdash_A A \leq B$, $A \downarrow_n A'$, and $B \downarrow_n B'$ then $\vdash_A A' \leq B'$.*

Proof By induction on derivations of $\vdash_A A \leq B$, using Church–Rosser for AWS-PROMOTE.

Lemma 4.7 (Normalization) *If $\vdash_A A \leq B$ then there are A' and B' such that $A \twoheadrightarrow_n A'$ and $B \twoheadrightarrow_n B'$.*

The following lemma simply states that promotion is always valid, even if the side condition of AWS-PROMOTE is not satisfied. This is true because if the side condition is not satisfied then AWS-TVAR can be applied.

Lemma 4.8 (Promotion) *If $\vdash_A B \leq C$, $A >_P B$ and there is a D such that $A \twoheadrightarrow_n D$ then $\vdash_A A \leq C$.*

Proof By Normalization there is a D such that $C \twoheadrightarrow_n D$. If $C \downarrow_n A$ then $\vdash_A A \leq C$ by AWS-TVAR, and otherwise $\vdash_A A \leq C$ by AWS-PROMOTE.

Lemma 4.9 (Transitivity) *If $\vdash_A A \leq B$ and $\vdash_A B \leq C$ then $\vdash_A A \leq C$.*

Proof By induction on derivations, using Normalization and Subject Conversion in AWS-TVAR and Promotion in AWS-PROMOTE.

The length of a derivation $T(A)$ includes uses of reduction. To prove Anti-Symmetry, we need a measure that is invariant under reduction but respects $>_P$.

Definition 4.10 *We define an alternative length measure $|T(A)|$ of a derivation of $T(A)$ inductively as:*

$$max(\{|T(B)| \text{ for } B \text{ such that } A \rhd B\} \cup \{|T(C)| + 1 \text{ for } C \text{ such that } A >_P C\})$$

Observe that $|T(A)|$ does not depend on $>_S$.

Lemma 4.11 *We have the following properties of the predicate $T(A)$ and the measure $|T(A)|$ of derivations of $T(-)$:*

1. *If $T(A)$ and $A \twoheadrightarrow_n B$ then $|T(A)| = |T(B)|$.*
2. *If $T(A)$, $T(B)$ and $A \downarrow_n B$ then $|T(A)| = |T(B)|$.*

3. *If* $T(A)$ *and* $A \rhd B$ *then* $|T(A)| = |T(B)|$.
4. $|T(\mathrm{T}_\star)| = 0$.
5. *If* $T(\Lambda X : K.A)$ *then* $|T(\Lambda X : K.A)| = 0$.
6. *If* $T(A_1 \to A_2)$ *then* $|T(A_1 \to A_2)| = 0$.
7. *If* $T(\forall X \leq A_1 : K.A_2)$ *then* $|T(\forall X \leq A_1 : K.A_2)| = 0$.

Lemma 4.12 (Key Lemma) *If* $T(A)$, $T(B)$ *and* $\vdash_A A \leq B$ *then* $|T(A)| \geq |T(B)|$.

Proof By induction on $\vdash_A A \leq B$, using Lemma 4.11.

Specifically, notice that the Key Lemma allows us to prove directly that $\vdash_A A(B_1,...,B_n) \leq X_A(B_1,...,B_n)$ is impossible, since $|T(X_A(B_1,...,B_n))| > |T(A(B_1,...,B_n))|$ by definition. We use this fact in the proof of Anti-Symmetry.

Lemma 4.13 (Anti-Symmetry) *If* $\vdash_A A \leq B$, $\vdash_A B \leq A$, $T(A)$ *and* $T(B)$, *then* $A \downarrow_n B$.

Proof By induction on derivations $\vdash_A A \leq B$ and $\vdash_A B \leq A$.

We consider two cases:

- AWS-PROMOTE is used in deriving $\vdash_A A \leq B$. This is a base case with no use of the induction hypothesis. By the Key Lemma $|T(A(B_1,...,B_n))| \geq |T(C)|$, and $|T(C)| \geq |T(X_A(B_1,...,B_n))|$, so $|T(A(B_1,...,B_n))| \geq |T(X_A(B_1,...,B_n))|$. However, $|T(X_A(B_1,...,B_n))| > |T(A(B_1,...,B_n))|$ by definition, which is a contradiction by trichotomy.
- AWS-TABS is used in deriving $\vdash_A A \leq B$ and $\vdash_A B \leq A$. Then $A \equiv \Lambda X : K.A_1$ and $B \equiv \Lambda X : K.B_1$ for some A_1 and B_1, with $\vdash_A A_1 \leq B_1$ and $\vdash_A B_1 \leq A_1$. By definition $\Lambda X : K.A_1 >_S A_1$ implies $T(A_1)$ and similarly $T(B_1)$, so by induction hypothesis $A_1 \downarrow_n B_1$, and so $\Lambda X : K.A_1 \downarrow_n \Lambda X : K.B_1$.

5 Completeness of the Algorithm

We now perform the logical relation proof to show completeness and decidability of the algorithm.

Definition 5.1 (Semantic Object) *A type* A *is a semantic object at kind* K, *written* $SO_K(A)$, *iff* $T(A)$ *and* $\vdash_A A \leq \mathrm{T}_K$.

As for typical strong normalization proofs, this notion of semantic object does not require well-formedness.

Definition 5.2 (Interpretation) *The interpretations of a kind* K, $\models A \in K$ *and* $\models A \leq B \in K$, *are defined by induction on* K:

- $\models A \in \star$ *iff* $SO_\star(A)$.
- $\models A \leq B \in \star$ *iff* $SO_\star(A)$, $SO_\star(B)$ *and* $\vdash_A A \leq B$.
- $\models A \in K \to K'$ *iff* $SO_{K \to K'}(A)$ *and* $\models A(B) \in K'$ *for all* B *such that* $\models B \in K$.
- $\models A \leq B \in K \to K'$ *iff* $SO_{K \to K'}(A)$, $SO_{K \to K'}(B)$, $\vdash_A A \leq B$ *and* $\models A(C) \leq B(C) \in K'$ *for all* C *such that* $\models C \in K$.

The interpretation extends to parallel substitutions $\models \gamma \in \Gamma$ *as follows:*

- $\models () \in ()$.

- $\models \gamma[X \leftarrow Y] \in \Gamma, X \leq A : K$ *iff* $\models \gamma \in \Gamma$.
- $\models \gamma[A/X] \in \Gamma, X : K$ *iff* $\models \gamma \in \Gamma$ *and* $\models A \in K$.

Observe that variables in the context of the form $X \leq A : K$ where $A \not\equiv \mathrm{T}_K$ can only take renamings $[X \leftarrow Y]$. These variables are never subject to substitution, since they cannot become the bound variable of an abstraction.

Lemma 5.3 (Saturated Sets) *The following properties hold for* $\models A \in K$ *and* $\models A \leq B \in K$*:*

1. *If* $\models A \leq B \in K$ *then* $\models A \in K$ *and* $\models B \in K$.
2. *If* $\models A \in K$ *then* $SO_K(A)$.
3. *If* $\models A \leq B \in K$ *then* $\vdash_A A \leq B$.
4. *If* $\models A \in K$ *then* $\models A \leq A \in K$.
5. $\models \mathrm{T}_K \in K$.
6. *If* $\models A \in K$ *then* $\models A \leq \mathrm{T}_K \in K$.
7. *If* $\models B \in K$ *and* $A >_P B$ *then* $\models A \in K$*, and similarly for the left- and right-hand sides of* $\models A \leq B \in K$.
8. *If* $\models B \in K$*,* $T(A)$ *and* $A \rightarrow_w B$ *then* $\models A \in K$*, and similarly for the left-hand side of* $\models A \leq B \in K$.
9. *If* $\models A' \leq B' \in K$*,* $A \downarrow_n A'$*,* $B \downarrow_n B'$*,* $T(A)$ *and* $T(B)$ *then* $\models A \leq B \in K$.
10. *If* $\models A \leq B \in K$ *and* $\models B \leq C \in K$ *then* $\models A \leq C \in K$.

Proof By induction on K, using for example Reflexivity for Case 4, Cases 2 and 4 for Case 6, Lemma 4.3 Case 9 and Promotion for Case 7, and Transitivity for Case 10.

Theorem 5.4 (Completeness) *Suppose* $\models \gamma \in \Gamma$*. Then:*

- *If* $\Gamma \vdash A : K$ *then* $\vdash A[\gamma] \in K$.
- *If* $\Gamma \vdash A \leq B : K$ *and* $\models A[\gamma] \leq B[\gamma] \in K$.

Proof By induction on derivations. We consider several cases.

- TopEmp. By Lemma 5.3 Case 5.
- TVar. If $\gamma = \gamma_0[X \leftarrow Y]$ then by induction hypothesis $\models B[\gamma] \in K$, and $X_{B[\gamma]} >_P B[\gamma]$, so $\models X_{B[\gamma]} \in K$ by Lemma 5.3 Case 7.
  If $\gamma = \gamma_0[A/X]$ then $\models A \in K$ by definition.
- TAbs. By definition $\models \gamma[X \leftarrow Y] \in \Gamma, X : K$, so by induction hypothesis $\models A[\gamma[X \leftarrow Y]] \in K'$, so $T(A[\gamma[X \leftarrow Y]])$ implies $T((\Lambda X : K.A)[\gamma] \equiv \Lambda Y : K.A[\gamma[X \leftarrow Y]])$ by Lemma 4.3.
  Furthermore, if $\models B \in K$ then $\models \gamma[B/X] \in \Gamma, X : K$ by definition. We have $\models A[\gamma[B/X]] \in K'$ by induction hypothesis and $T(A[\gamma[B/X]])$ by definition. Then $T((\Lambda X : K.A)[\gamma])$ above and $T(B)$ by Lemma 5.3, and $(\Lambda X : K.A)[\gamma](B) \rightarrow_w A[\gamma[B/X]]$, so $T((\Lambda X : K.A)[\gamma](B))$ by Lemma 4.3 Case 11. Therefore, $\models (\Lambda X : K.A)[\gamma](B) \in K'$, and so $\models (\Lambda X : K.A)[\gamma] \in K \rightarrow K'$.
- $\forall$. By induction hypothesis $\models A[\gamma] \in K$, and $\models \gamma[X \leftarrow Y] \in \Gamma, X \leq A : K$ implies $\models B[\gamma[X \leftarrow Y]] \in \star$ by induction hypothesis, so $T(A[\gamma])$ and $T(B[\gamma[X \leftarrow Y]])$ by Lemma 5.3. Then $T((\forall X \leq A : K.B)[\gamma] \equiv \forall X \leq A[\gamma] : K.B[\gamma[X \leftarrow Y]])$ by Lemma 4.3, so $\models (\forall X \leq A : K.B)[\gamma] \in \star$ by definition.
- S-Refl. By induction hypothesis $\models A \in K$, so by Lemma 5.3 Case 4 $\models A \leq A \in K$.

- S-TRANS. By induction hypothesis and Lemma 5.3 Case 10.
- S-TOP. By induction hypothesis and Lemma 5.3 Case 6.
- S-TAPP. By induction hypothesis $\models A[\gamma] \leq C[\gamma] \in K \to K'$ and $\models B[\gamma] \leq D[\gamma] \in K$ and $\models D[\gamma] \leq B[\gamma] \in K$. Then $B[\gamma] \downarrow_n D[\gamma]$ by Lemma 5.3 Case 2 and Anti-Symmetry, and by definition $\models (A(B))[\gamma] \equiv (A[\gamma])(B[\gamma]) \leq (C[\gamma])(B[\gamma]) \equiv (C(B))[\gamma] \in K'$. We also know $\models C[\gamma] \in K \to K'$ by Lemma 5.3 and $\models D[\gamma] \in K$, so $\models (C(D))[\gamma] \equiv (C[\gamma])(D[\gamma]) \in K'$ by definition and $SO_{K'}((C(D))[\gamma])$ by Lemma 5.3 Case 2. Therefore $(C[\gamma])(B[\gamma]) \downarrow_n (C[\gamma])(D[\gamma])$ implies $\models (A(B))[\gamma] \leq (C(D))[\gamma] \in K'$ by Lemma 5.3 Case 9.

Lemma 5.5 $\models \mathrm{id}_\Gamma \in \Gamma$.

Proof By induction on Γ.

Corollary 5.6 (Termination) *If* $\Gamma \vdash A \leq B : K$ *then* $T(A)$ *and* $T(B)$*, and* $\vdash_A A \leq B$.

Corollary 5.7 (Anti-Symmetry) *If* $\Gamma \vdash A \leq B : K$ *and* $\Gamma \vdash B \leq A : K$ *then* $A \downarrow_n B$.

Corollary 5.8 *If* $\Gamma \vdash \mathrm{T}_K \leq A : K$ *then* $A \downarrow_n \mathrm{T}_K$.

6 Correctness

So far we have not needed any properties of the judgements $\Gamma \vdash J$. We now develop some metatheory for those judgements and use the results to prove the correctness of the algorithm.

Lemma 6.1 (Context) 1. *If* $\Gamma \vdash J$ *then* $\mathrm{FV}(J) \subseteq \mathrm{dom}(\Gamma)$.

2. *If* $X \leq A : K \in \Gamma$ *and* $\Gamma \vdash \mathrm{ok}$ *then* $X \notin \mathrm{FV}(A)$.
3. *If* $\Gamma \vdash J$ *then* $\Gamma \vdash \mathrm{ok}$ *as a sub-derivation.*
4. *If* $\Gamma, \Gamma' \vdash \mathrm{ok}$ *then* $\Gamma \vdash \mathrm{ok}$.

Definition 6.2 (Renaming) γ *is a renaming for* Γ *in* Δ *if* $\Delta \vdash \mathrm{ok}$*,* γ *is a renaming, and* $\gamma(X) \leq A[\gamma] : K \in \Gamma$ *for each* $X \leq A : K \in \Gamma$.

Lemma 6.3 (Renaming) *If* $\Gamma \vdash J$ *and* γ *is a renaming for* Γ *in* Δ *then* $\Delta \vdash J[\gamma]$.

Lemma 6.4 *If* $\Gamma, \Gamma' \vdash \mathrm{ok}$*,* $\Gamma \vdash A : K$ *and* $X \notin \mathrm{dom}(\Gamma, \Gamma')$ *then* $\Gamma, X \leq A : K, \Gamma' \vdash \mathrm{ok}$.

Proposition 6.5 (Replacement) *If* $\Gamma, X \leq B : K, \Gamma' \vdash J$*,* $\Gamma \vdash A \leq B : K$ *and* $\Gamma \vdash A : K$ *then* $\Gamma, X \leq A : K, \Gamma' \vdash J$.

Proposition 6.6 (Thinning) *If* $\Gamma, \Gamma' \vdash J$*,* $\Gamma \vdash A : K$ *and* $X \notin \mathrm{dom}(\Gamma, \Gamma')$ *then* $\Gamma, X \leq A : K, \Gamma' \vdash J$.

Proof By Lemmas 6.1 and 6.4 $\Gamma, X \leq A : K, \Gamma' \vdash \mathrm{ok}$. Observe that id_Γ is a renaming for Γ, Γ' in $\Gamma, X \leq A : K, \Gamma'$. Then $\Gamma, X \leq A : K, \Gamma' \vdash J$ by Renaming.

Proposition 6.7 (Substitution) *If* $\Gamma, X : K, \Gamma' \vdash J$ *and* $\Gamma \vdash A : K$ *then* $\Gamma, \Gamma'[A/X] \vdash J[A/X]$.

Lemma 6.8 (Subject Reduction) *If* $\Gamma \vdash A : K$ *and* $A \rhd B$ *then* $\Gamma \vdash A = B : K$.

Proof By induction on derivations.
Add generation.

Proposition 6.9 (Correctness) *The algorithm is correct for the declarative judgements:*

- *If* $\Gamma \vdash$ *ok and* $\Gamma \vdash_A A : K$ *then* $\Gamma \vdash A : K$.
- *If* $\Gamma \vdash A, B : K$ *and* $\vdash_A A \leq_W B$ *then* $\Gamma \vdash A \leq B : K$.
- *If* $\Gamma \vdash A, B : K$ *and* $\vdash_A A \leq B$ *then* $\Gamma \vdash A \leq B : K$.

Proof By induction on derivations, using Context and Renaming in AT-TVAR; the generation property and Subject Reduction for AWS-TVAR; the generation property for AWS-TOP and AWS-PROMOTE; Subject Reduction and Context Replacement in AWS-ALL, and Subject Reduction for AS-INC.

Corollary 6.10 (Decidability of Subtyping) *Subtyping is decidable.*

Proof Suppose $\Gamma \vdash A, B : K$. By Corollary 5.6 $T(A)$ and $T(B)$, and so $\vdash_A A \leq B$ is decidable by Proposition 4.4, and so by Correctness, $\Gamma \vdash A \leq B : K$ is also decidable.

7 Relationship with Traditional $F^{\omega}_{\leq}$

We now show that the system with the bounded variable constructor is equivalent to the traditional presentation of the system, without bounds in the variable constructor.

Explicitly, we take the syntax of traditional $F^{\omega}_{\leq}$ [5, 12], which differs from the syntax presented here only by having a type constructor X instead of the bounded type constructor X_A. We write judgements in this system as $\Gamma \vdash_T J$, with the decoration T for traditional. The rules of inference are also standard: we include two rules here but refer the reader to the standard references for the complete system.[1]

$$\frac{\Gamma \vdash_T A, B : K \qquad A =_\beta B}{\Gamma \vdash_T A \leq B} \tag{TS-CONV}$$

$$\frac{\Gamma \vdash_T A \leq B \qquad \Gamma \vdash_T A(C) : K}{\Gamma \vdash_T A(C) \leq B(C)} \tag{TS-TAPP}$$

Definition 7.1 (Decoration) *Decoration of a type,* A^Γ*, where* Γ *is a context in the system with bounded variables, is a partial function that maps types from the Curry-style presentation to the more explicit structure à la Church:*

- $X^\Gamma = X_B$*, if* $X \leq B : K \in \Gamma$.
- $(A \to B)^\Gamma = A^\Gamma \to B^\Gamma$.
- $(\forall X \leq A : K.B)^\Gamma = \forall X \leq A^\Gamma : K.B^{\Gamma, X \leq A:K}$.
- $(\Lambda X : K.A)^\Gamma = \Lambda X : K.A^{\Gamma, X:K}$.
- $(A(B))^\Gamma = A^\Gamma(B^\Gamma)$.
- $(\mathrm{T}_\star)^\Gamma = \mathrm{T}_*$.

The extension to contexts, Γ^d*, is defined in the obvious way:*

- $()^d = ()$.
- $(\Gamma, X \leq A : K)^d = \Gamma^d, X \leq A^{\Gamma^d} : K$.

[1] However, [5] has intersection types rather than an explicit rule for $\mathrm{T}_\star$.

Definition 7.2 (Erasure) *The erasure map is simply the homomorphic extension of the stripping of the bound from the variable constructor:*

$$|X_A| \quad = \quad X$$

The extension to contexts is also the homomorphic extension.

Decoration and erasure have some simple properties, most important of which is that both preserve β-equality:

Lemma 7.3
- *If $A =_\beta B$ then $|A| =_\beta |B|$.*
- *If $A =_\beta B$ and A^Γ, B^Γ defined then $A^\Gamma =_\beta B^\Gamma$.*

Now, we can relate the Curry and Church presentations of $F^\omega_\leq$. The proofs rely on standard properties of the traditional presentation, for example Church–Rosser for untyped reduction, Generation properties, well-formedness of contexts, well-kindedness of subtyping, and uniqueness of kinds.

Lemma 7.4 (Soundness)
- *If $\Gamma \vdash_T A : K$ then Γ^d and A^{Γ^d} are defined and $\Gamma^d \vdash A^{\Gamma^d} : K$.*
- *If $\Gamma \vdash_T A \leq B$ and $\Gamma \vdash_T A, B : K$ then Γ^d, A^{Γ^d}, B^{Γ^d} are well-formed and $\Gamma^d \vdash A^{\Gamma^d} \leq B^{\Gamma^d} : K$.*

Proof We consider the rule TS-CONV. By Church–Rosser, there exists a C such that $A \rhd^* C$ and $B \rhd^* C$. By Lemma 7.3, $A^{\Gamma^d} \rhd^* C^{\Gamma^d}$ and $B^{\Gamma^d} \rhd^* C^{\Gamma^d}$. By the induction hypothesis, $\Gamma^d \vdash A^{\Gamma^d} : K$ and $\Gamma^d \vdash B^{\Gamma^d} : K$. By Subject Reduction, $\Gamma^d \vdash A^{\Gamma^d} = C^{\Gamma^d} : K$ and $\Gamma^d \vdash B^{\Gamma^d} = C^{\Gamma^d} : K$, and we have $\Gamma^d \vdash A^{\Gamma^d} \leq C^{\Gamma^d} : K$ and $\Gamma^d \vdash C^{\Gamma^d} \leq B^{\Gamma^d} : K$ by definition. Finally, by S-TRANS, $\Gamma^d \vdash A^{\Gamma^d} \leq B^{\Gamma^d} : K$.

Lemma 7.5 (Completeness)
- *If $\Gamma \vdash A : K$ then $|\Gamma| \vdash_T |A| : K$.*
- *If $\Gamma \vdash A \leq B : K$ then $|\Gamma| \vdash_T |A| \leq |B|$ and $|\Gamma| \vdash_T |A|, |B| : K$.*

Proof We consider the rule S-TAPP. By the induction hypothesis, $|\Gamma| \vdash_T |A| \leq |C|$, $|\Gamma| \vdash_T |A|, |C| : K \to K'$, $|\Gamma| \vdash_T |B| \leq |D|$, and $|\Gamma| \vdash_T |B|, |D| : K$. By the kinding rule for type application in the traditional presentation and the definition of erasure, we have $|\Gamma| \vdash_T |AB|, |CB|, |CD| : K'$, and by TS-TAPP $|\Gamma| \vdash_T |AB| \leq |CB|$. By Anti-Symmetry $B =_\beta D$, and $|B| =_\beta |D|$ by Lemma 7.3, so $|CB| =_\beta |CD|$. By TS-CONV $|\Gamma| \vdash_T |CB| \leq |CD|$, and finally, by Transitivity, $|\Gamma| \vdash_T |AB| \leq |CD|$.

The important metatheoretic results for subtyping now transfer straightforwardly to the traditional one.

Corollary 7.6
- *If $\Gamma \vdash_T A : K$ then A is strongly normalizing.*
- *Typechecking $\Gamma \vdash_T A : K$ and subtyping $\Gamma \vdash_T A \leq B : K$ are decidable.*
- *If $\Gamma \vdash_T A \leq B$, $\Gamma \vdash_T B \leq A$, $\Gamma \vdash_T A : K$ and $\Gamma \vdash_T B : K$ then $A =_\beta B$.*

8 Related and Future Work

An earlier version of this article was published in the unrefereed proceedings of Henk Barendregt's Festschrift [9]. The current paper extends the results of the earlier version by showing the equivalence with the traditional presentation of the system.

In an earlier paper [7], we considered an algorithm that reduces types to normal form before invoking the promotion rule in the algorithm. This makes context replacement trivial for equal types, since they have the same normal form and so altering the context does not alter the path of types considered by the algorithm. However, this algorithm is not optimal, since it normalizes the head earlier than necessary.

That earlier paper also used a typed operational semantics to show termination of the algorithm. This gave a more extensive treatment of the metatheory, and the admissibility of thinning, substitution and context replacement were consequences of the model. Furthermore, Subject Reduction was straightforward in the typed operational semantics. In the current paper, finding the exact formulation necessary to show these results in the declarative system $\Gamma \vdash J$ turned out to be somewhat subtle, since the kinding judgement uses the subtyping judgement for the bounded variable rule. However, the approach using typed operational semantics was also longer and less approachable, and involved Kripke models for the proof of completeness. We hope that the current paper is clearer by not defining an intermediate system.

In separate work [8], we also proved anti-symmetry of higher-order subtyping using the typed operational semantics. The basic idea of that paper was to include the sub-derivation of replacing the variable in a bounded head variable expression $X_A(B_1,...,B_n)$ with its bound, $A(B_1,...,B_n)$. This idea is captured in the current paper by the $T(-)$ predicate. The $T(-)$ predicate is also similar to Compagnoni's approach with $+$-reduction [5], but we do not need to develop the metatheory of a new reduction relation.

As mentioned in the introduction, Stone and Harper [13] use a logical relation defined over sets of contexts, instead of the standard logical relations over single contexts, to show termination of an algorithm for a type theory with singleton types, Σ and Π types, and all of the η rules. Their work does not normalize the singleton types. This is an elegant solution to the problem of varying contexts, but it raises the question of why singletons or $F^{\omega}_{\leq}$ should have different requirements on the Kripke-style relation than other type systems.

Abel [1] has shown equivalence of a subtyping algorithm for higher kinds with polarity by direct induction on kinds rather than using a logical relation.

There are several directions for future work. We would like to show that a Harper–Pfenning-style algorithm [10] is correct and complete for the type system. Furthermore, it would be nice to be able to prove context conversion and Church–Rosser in the model, as can be done for logical relations for equality, rather than proving them for the algorithm and lifting to the model. However, properties that follow straightforwardly for equality, such as that $\models A = B \in K$ implies $\models A = A \in K$, cannot be shown so easily for subtyping. Finally, another candidate type construct that we might study with our technique of explicit type information is singleton types, which also have computational behavior expressed in the context.

9 Conclusions

We have introduced a natural and powerful extension of the syntax of $F^{\omega}_{\leq}$ and showed that the development of the metatheory is similar to the standard metatheory for type theories, specifically without a Kripke-style model and with a simple inductive definition capturing termination of the algorithm. We have shown all of the important results for the system, including anti-symmetry, transitivity elimination and decidability of subtyping.

References

[1] Andreas Abel (2008): *Polarized Subtyping for Sized Types*. *Mathematical Structures in Computer Science* 18(5), pp. 797–822.

[2] Henk Barendregt (1992): *Lambda Calculi with Types*. In: *Handbook of Logic in Computer Science, Volumes 1 (Background: Mathematical Structures) and 2 (Background: Computational Structures), Abramsky*

& Gabbay & Maibaum (Eds.), Clarendon. 2, Oxford University Press. Available at `citeseer.ist.psu.edu/barendregt92lambda.html`.

[3] Luca Cardelli (1990): *Notes about $F^{\omega}_{<:}$*. Unpublished manuscript.

[4] Luca Cardelli & Giuseppe Longo (1991): *A semantic basis for Quest*. *Journal of Functional Programming* 1(4), pp. 417–458.

[5] Adriana Compagnoni (1995): *Higher-Order Subtyping with Intersection Types*. Ph.D. thesis, University of Nijmegen.

[6] Adriana Compagnoni & Healfdene Goguen (1997): *Decidability of Higher-Order Subtyping via Logical Relations*. Available at `ftp://www.dcs.ed.ac.uk/pub/hhg/hosdec.ps.gz`. A later version is published as [7].

[7] Adriana Compagnoni & Healfdene Goguen (2003): *Typed Operational Semantics for Higher-Order Subtyping*. *Information and Computation* 184(2), pp. 242–297.

[8] Adriana Compagnoni & Healfdene Goguen (2006): *Anti-symmetry of Higher-Order Subtyping*. *Mathematical Structures in Computer Science* 16(1), pp. 41–65.

[9] Adriana Compagnoni & Healfdene Goguen (2007): *Subtyping à la Church*. In Erik Barendsen, Venanzio Capretta, Herman Geuvers & Milad Niqui, editors: *Reflections on Type Theory, λ-calculus, and the Mind. Essays dedicated to Henk Barendregt on the Occasion of his 60th Birthday*. Radboud University Nijmegen.

[10] Robert Harper & Frank Pfenning (2005): *On equivalence and canonical forms in the LF type theory*. *ACM Trans. Comput. Logic* 6(1), pp. 61–101, doi:http://doi.acm.org/10.1145/1042038.1042041.

[11] John C. Mitchell (1990): *Toward a Typed Foundation for Method Specialization and Inheritance*. In: *Proceedings of the 17th ACM Symposium on Principles of Programming Languages*. pp. 109–124.

[12] Martin Steffen & Benjamin Pierce (1997): *Higher-Order Subtyping*. *Theoretical Computer Science* 176(1–2), pp. 235–282. Corrigendum in TCS vol. 184 (1997), p. 247.

[13] Christopher A. Stone & Robert Harper (2006): *Extensional equivalence and singleton types. ACM Trans. Comput. Log.* 7(4), pp. 676–722. Available at `http://doi.acm.org/10.1145/1183278.1183281`.

[14] Thomas Streicher (1991): *Semantics of Type Theory: Correctness, Completeness and Independence Results*. Birkhäuser.