SUBTYPING À LA CHURCH

ADRIANA COMPAGNONI AND HEALFDENE GOGUEN

Stevens Institute of Technology
e-mail address: abc@cs.stevens.edu

Google
e-mail address: hhg@google.com

Abstract. Type theories with higher-order subtyping or singleton types are examples of systems where the computational behavior of variables is determined 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 simple 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. Completeness and correctness, anti-symmetry, transitivity elimination and termination of the algorithm are all proved.

In Honor of Henk Barendregt on the Occasion of his 60th Birthday

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 PTS's [Bar92], 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 the computational behavior of variables can change according to context information. The key difficulty is that strong normalization of a term depends on information in the context, but that normalization also needs to be closed under replacement by equal contexts, in order to model the term 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 uses of promotion, or replacing a

2000 ACM Subject Classification: F.3.2, F.3.3, F.4.1.
Key words and phrases: Lambda Calculus, Subtyping, Kripke Model.

variable by its bound. 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 [CG03, CG97] 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 [SH06] prove termination for an algorithm for singletons using the unnormalized singleton rather than normalizing it first. Their Kripke-style proof indexes the 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 $F^\omega_\leq$ [Car90, CL91, Mit90], where the type constructor for variables is X_A with the variable's bound A explicitly mentioned. 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 it is never necessary to promote a variable to a type convertible with its bound, the cause of the difficulties in the system without bounded variables. 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.

The idea of introducing additional information in the term structure is not new. We call this presentation "à la Church" for its obvious similarity to type labels on λ-abstractions. Streicher [Str91] adds type labels to applications and abstractions in the Calculus of Constructions in order to define a partial interpretation function, which is subsequently proved total. Barbanera and Berardi [BB95] use type labels, including on variables, to guarantee well-typedness of terms under classical reduction. The second author [Gog05] used type labels on variables in order to define a syntactic translation on terms to show the decidability of $\beta\eta$-equality based on strong normalization for β-reduction.

There are several complications resulting from the bounded variable constructor. First, in order to show correctness of the traditional system without explicitly bounded variables, we would need to show the equivalence of the two presentations. This is left as future work. Secondly, because variables mention their bound explicitly, we need to distinguish between 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. Finally, the subtyping judgement is needed in the formulation of the inference rules for the kinding judgement: this is not necessary in the traditional presentation, for example in the rule TVar for kinding a type variable.

The focus of this paper is on the properties of the subtyping relation, including completeness and correctness, anti-symmetry, transitivity elimination and decidability. 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.

The structure of the rest of the paper is as follows. In Section 2 we introduce the term syntax and the judgements and rules of inference for $F^{\omega}_{\leq}$. In Section 3 we introduce the algorithm, and in Section 4 we study basic properties of the algorithm such as decidability and its being a partial order. In Section 5 we show completeness of the algorithm, and in Section 6 we show correctness. Finally, in Section 7 we discuss related and future work, and in Section 8 we conclude.

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 $X, Y, Z, ...$. The types include variables with explicit bounds X_A; the top type $\mathrm{T}_\star$; function types $A \to B$; and polymorphic types $\forall X \leq A : K.B$, 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 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.

We identify types that differ only in the names of bound variables. We write $A(B_1, ..., B_n)$ for $(A(B_1))...(B_n)$. If $A \equiv X_C(B_1, ..., B_n)$ 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 $\mathrm{T}_{K \to K'} = \Lambda X : K.\mathrm{T}_{K'}$.

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 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 $\Gamma \vdash \mathrm{T}_\star : \star$, 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. *Admissible Rules.* The following rules can be shown admissible by induction on derivations.

$$\frac{\Gamma, X \leq B : K, \Gamma' \vdash J \qquad \Gamma \vdash A \leq B : K \qquad \Gamma \vdash A : K}{\Gamma, X \leq A : K, \Gamma' \vdash J} \quad (\textsc{Repl})$$

$$\frac{\Gamma, \Gamma' \vdash J \qquad \Gamma \vdash A : K \qquad X \notin \mathrm{dom}(\Gamma, \Gamma')}{\Gamma, X \leq A : K, \Gamma' \vdash J} \quad (\textsc{Thin})$$

$$\frac{\Gamma, X : K, \Gamma' \vdash J \qquad \Gamma \vdash A : K}{\Gamma, \Gamma'[A/X] \vdash J[A/X]} \quad (\textsc{Subst})$$

2.2.2. *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.

$$() \vdash \mathrm{T}_\star : \star \quad (\textsc{TopEmp})$$

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

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

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

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

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

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

Notice that in rule TVar it is possible that $X \in B$ when $\Gamma \vdash X_B : K$.

2.2.3. *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.

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

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

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

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

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

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

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

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

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

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

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

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* $A \to_w C$.
- $X_A(B_1, ..., B_n) \twoheadrightarrow_n X_C(D_1, ..., D_n)$ *if* $A \twoheadrightarrow_n C$ *and* $B_i \twoheadrightarrow_n D_i$ *for* $1 \leq i \leq n$.
- $\Lambda X : K.A \twoheadrightarrow_s \Lambda X : K.B$ *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* $B \twoheadrightarrow_n C$.

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 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. Since it incorporates weak-head reduction, it is clearly not terminating in general.

$$\Gamma \vdash_A \mathrm{T}_\star : \star \qquad \text{(AT-Top)}$$

$$\frac{X \leq A : K \in \Gamma}{\Gamma \vdash_A X_A : K} \qquad \text{(AT-TVar)}$$

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

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

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

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

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

$$\frac{A \downarrow_n C \qquad B_i \downarrow_n D_i \ (1 \leq i \leq n)}{\vdash_A X_A(B_1, ..., B_n) \leq_W X_C(D_1, ..., D_n)} \qquad \text{(AWS-TVar)}$$

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

$$\frac{\vdash_A A \leq B}{\vdash_A \Lambda X : K.A \leq_W \Lambda X : K.B} \qquad \text{(AWS-TAbs)}$$

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

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

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

The rule for subtyping of bounded variables, S-TVar, states that X_A is less than X_B if A and B are equal. It is tempting to instead allow A to be a subtype of B, but it appears that the algorithm cannot be defined for this system. For completeness, we would require a promotion rule, in the base case for bounded variables, that X_A be a subtype of B if A is a subtype of B and the weak-head normal form of B is not X_C with A a subtype of C. This last condition has the subtyping statement in a negative position, and therefore is not a valid inductive definition.

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 a 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 and 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.

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. □

Lemma 4.4 (Reflexivity)**.** *If $T(A)$ then $\vdash_A A \leq A$.*

Proof. By induction on $T(A)$, we show that if A is weak-head normal then $\vdash_A A \leq_W A$ and otherwise $\vdash_A A \leq A$.

If A is not weak-head normal then $A \twoheadrightarrow_w B$ and $\vdash_A B \leq_W B$ by induction hypothesis.

Otherwise, 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 $\vdash_A X_B(C_1, ..., C_n) \leq_W X_B(C_1, ..., C_n)$. □

Lemma 4.5 (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 4.6 (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 4.7 (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 4.8 (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 4.9. *We define the length $|T(A)|$ of a derivation of $T(A)$ inductively as the maximum of $|T(B)|$ for $A \rhd B$ and $|T(B)|+1$ for $A >_P B$.*

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

Lemma 4.10. *We have the following properties of the predicate $T(A)$ and the length $|T(A)|$ 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 4.11 (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 4.10. □

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 4.12 (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 $\vdash_A A \leq B$. 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.
- 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 $\Lambda X : K.A_1 \downarrow_n \Lambda X : K.B_1$.

□

Proposition 4.13 (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.

□

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 $\vdash_A A \leq \mathrm{T}_K$.*

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

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.
- TApp. By induction hypothesis $\models A[\gamma] \in K \rightarrow K'$ and $\models B[\gamma] \in K$, so $\models (A(B))[\gamma] \equiv (A[\gamma])(B[\gamma]) \in K'$ 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 $T((\Lambda X : K.A)[\gamma] \equiv \Lambda Y : K.A[\gamma[X \leftarrow Y]])$.
  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 $T((\Lambda X : K.A)[\gamma])$ by Lemma 4.3 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 $\models (\Lambda X : K.A)[\gamma] \in K \rightarrow K'$.
- Arrow. By induction hypothesis $\models A[\gamma], B[\gamma] \in \star$, so $\models (A \rightarrow B)[\gamma] \equiv A[\gamma] \rightarrow B[\gamma] \in \star$ by Lemma 4.3 and definition.
- $\forall$. 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 $\models A[\gamma] \le C[\gamma] \in K \to K'$ and $\models B[\gamma] \le D[\gamma] \in K$ and $\models D[\gamma] \le B[\gamma] \in K$. Then $B[\gamma] \downarrow_n D[\gamma]$ by Lemma 5.3 Case 2 and Anti-Symmetry, and by definition $\models (A(B))[\gamma] \equiv (A[\gamma])(B[\gamma]) \le (C[\gamma])(B[\gamma]) \equiv (C(B))[\gamma] \in K'$. 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 $\models (A(B))[\gamma] \le (C(D))[\gamma] \in K'$ by Lemma 5.3 Case 9.
- S-Arrow. By induction hypothesis $\models B_1[\gamma] \le A_1[\gamma] \in \star$ and $\models A_2[\gamma] \le B_2[\gamma] \in \star$. By Lemma 5.3 Case 2 and Lemma 4.3 Case 7 we have $T((A_1 \to A_2)[\gamma])$ and $T((B_1 \to B_2)[\gamma])$, and so $\models (A_1 \to A_2)[\gamma], (B_1 \to B_2)[\gamma] \in \star$. Furthermore, by Lemma 5.3 Case 2 $\vdash_A B_1[\gamma] \le A_1[\gamma]$ and $\vdash_A A_2[\gamma] \le B_2[\gamma]$, and so $\vdash_A (A_1 \to A_2)[\gamma] \le (B_1 \to B_2)[\gamma]$ by AWS-Arrow, and $\models (A_1 \to A_2)[\gamma] \le (B_1 \to B_2)[\gamma] \in \star$ by definition.

□

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

Proof. By induction on Γ. □

Corollary 5.6 (Termination). *If $\Gamma \vdash A \le B : K$ then $T(A)$ and $\mathrm{SN}(A)$, $T(B)$ and $\mathrm{SN}(B)$, and $\vdash_A A \le B$.*

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

Corollary 5.8. *If $\Gamma \vdash \mathrm{T}_K \le A : K$ 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 $X \le A : K \in \Gamma$ 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 $\gamma(X) \le A[\gamma] : K \in \Gamma$ for each $X \le A : K \in \Gamma$.*

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 $\Gamma, X \le A : K, \Gamma' \vdash \mathrm{ok}$.*

Proposition 6.5 (Thinning). *If $\Gamma, \Gamma' \vdash J$, $\Gamma \vdash A : K$ and $X \notin \mathrm{dom}(\Gamma, \Gamma')$ then $\Gamma, X \le A : K, \Gamma' \vdash J$.*

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

Proposition 6.6 (Substitution)**.** *If* $\Gamma, X : K, \Gamma' \vdash J$ *and* $\Gamma \vdash A : K$ *then* $\Gamma, \Gamma'[A/X] \vdash J[A/X]$.

Lemma 6.7 (Subject Reduction)**.** *If* $\Gamma \vdash A : K$ *and* $A \rhd B$ *then* $\Gamma \vdash A = B : K$.

Proof. By induction on derivations. □

Proposition 6.8 (Correctness)**.** *The algorithm is correct for the kinding judgement:*

- *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 6.9 (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 4.13, and so by Correctness, $\Gamma \vdash A \leq B : K$ is also decidable. □

7. Related and Future Work

In an earlier paper [CG03], 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 [CG06], we also proved anti-symmetry of higher-order subtyping using the typed operational semantics. The basic idea of that paper was to include the subderivation 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 [Com95], but we do not need to develop the metatheory of a new reduction relation.

As mentioned in the introduction, Stone and Harper [SH06] 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.

There are several directions for future work. We intend to relate the system to the traditional system without explicit bounds on type variables. We would also like to show that a Harper-Pfenning-style algorithm [HP05] is correct and complete for the type system. Finally, 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.

8. Conclusions

We have introduced a simple but 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.

Acknowledgments

We thank the organizers of Henk Barendregt's Festschrift for inviting Adriana to submit an article.

References

[Bar92] Henk Barendregt. 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*, volume 2. Oxford University Press, 1992.

[BB95] Franco Barbanera and Stefano Berardi. A strong normalization result for classical logic. *Annals of Pure and Applied Logic*, 76(2):99–116, 1995.

[Car90] Luca Cardelli. Notes about $\mathrm{F}^\omega_{<:}$. Unpublished manuscript, October 1990.

[CG97] Adriana Compagnoni and Healfdene Goguen. Decidability of higher-order subtyping via logical relations, December 1997. A later version is published as [CG03].

[CG03] Adriana Compagnoni and Healfdene Goguen. Typed operational semantics for higher-order subtyping. *Information and Computation*, 184(2):242–297, August 2003.

[CG06] Adriana Compagnoni and Healfdene Goguen. Anti-symmetry of higher-order subtyping. *Mathematical Structures in Computer Science*, 16(1):41–65, February 2006.

[CL91] Luca Cardelli and Giuseppe Longo. A semantic basis for Quest. *Journal of Functional Programming*, 1(4):417–458, October 1991.

[Com95] Adriana Compagnoni. *Higher-Order Subtyping with Intersection Types.* PhD thesis, University of Nijmegen, 1995.

[Gog05] Healfdene Goguen. A syntactic approach to eta equality in type theory. In *ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages*, pages 75–84, January 2005.

[HP05] Robert Harper and Frank Pfenning. On equivalence and canonical forms in the LF type theory. *ACM Trans. Comput. Logic*, 6(1):61–101, 2005.

[Mit90] John C. Mitchell. Toward a typed foundation for method specialization and inheritance. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages*, pages 109–124, January 1990.

[SH06] Christopher A. Stone and Robert Harper. Extensional equivalence and singleton types. *ACM Trans. Comput. Log.*, 7(4):676–722, 2006.

[Str91] Thomas Streicher. *Semantics of Type Theory: Correctness, Completeness and Independence Results.* Birkhäuser, 1991.