Higher-Order Subtyping

Martin Steffen* Benjamin Pierce†

February 2, 1994

University of Edinburgh
Technical Report ECS-LFCS-94-280
and
IMMD VII, Universität Erlangen-Nürnberg
Interner Bericht IMMD7-01/94

Revised version with Fun subtyping

Abstract

System $F^{\omega}_{\leq}$ is an extension with subtyping of Girard's higher-order polymorphic λ-calculus. We develop the fundamental metatheory of this calculus: decidability of β-conversion on well-kinded types, elimination of the "cut-rule" of transitivity from the subtype relation, and the soundness, completeness, and termination of algorithms for subtyping and typechecking.

Keywords: lambda-calculus, type systems, subtyping, polymorphism, bounded quantification, typechecking.

1 Introduction

Since the early 1980's, increasing attention in the programming language community has been devoted to formal models for statically typed object-oriented languages. Cardelli [Car84] observed that refinement of object interfaces can be modeled by records and a simple form of *subtyping*. To account for the types of message-sending operations, Cardelli and Wegner [CW85] introduced *bounded quantification*. Fully capturing the object model of languages like Smalltalk required one further refinement, the extension to calculi with *higher-order polymorphism*, to deal properly with the interaction between subtyping and object encapsulation.

A number of typed object models have been given in this general setting.[1] Cook, Canning, Hill, Olthoff, and Mitchell [CCH+89, CHC90] proposed a variant called *F-bounded quantification*, which was used by Bruce [Bru93] to give the first full account of static typing for Smalltalk-style objects. Pierce and Turner [PT93a, PT93b] gave a similar model using existential types instead of recursive types to capture object encapsulation, effectively working within pure $F^{\omega}_{\leq}$. These two approaches were generalized by Hofmann and Pierce [HP94] to an abstract, axiomatic presentation of objects and subtyping.

---

*University of Erlangen-Nürnberg, IMMD VII, Martensstr. 3, 91058 Erlangen, Germany. Electronic mail: mnsteffe@informatik.uni-erlangen.de.

†Department of Computer Science, University of Edinburgh, The King's Buildings, Edinburgh, EH9 3JZ, U.K. Electronic mail: bcp@dcs.ed.ac.uk.

[1] Models for various object-oriented features have also been given using different techniques [MHF93, Aba93, CGL92, Cas92, Car92, AC94b, AC94a, Mit90, Wan87, Wan88, Wan89, etc.].

The core calculus underlying all of these models is an extension of Girard's higher-order polymorphic λ-calculus, System F^ω, with subtyping. Many of the ideas behind this system, called $F^\omega_\leq$ ("F-omega-sub"), are due to Cardelli, particularly to his 1988 paper, "Structural Subtyping and the Notion of Power Type" [Car88]; the extension of the subtype relation to type operators was developed by Cardelli and Mitchell [Car90, Mit90, BM92]. An early semantic model was given by Cardelli and Longo [CL91] using partial equivalence relations. Compagnoni and Pierce [CP93] gave a model for an extension of $F^\omega_\leq$ with intersection types. A more powerful model including recursive types was given by Bruce and Mitchell [BM92].

The second order fragment of $F^\omega_\leq$, called $F_\leq$, has been studied in detail, yielding both positive [Mar88, BL90, BCGS91, Ghe90, CG92, CG91, CMMS91, Ghe93b] and some surprising negative results [Ghe93a, Ghe93b, GP92, Pie92] including undecidability. Decidable variants of $F_\leq$ have been proposed [CW85, KS92, CP94] (our formulation of $F^\omega_\leq$ is based on [CW85]). But for the full ω-order calculus, next to nothing is known.[2]

The analysis of $F^\omega_\leq$ is significantly more challenging than that of $F_\leq$, principally because $F^\omega_\leq$ introduces a rule of *conversion* guaranteeing that β-convertible types occupy the same equivalence class in the subtype relation. This rule interacts with the rule of transitivity, requiring a substantial generalization of the standard cut-elimination argument — a key step in the proof of decidability, where uses of transitivity are restricted to a well-behaved form. Another significant difficulty is showing the termination of the final algorithm; in decidable variants on $F_\leq$, this is fairly straightforward; here, the proof depends on the strong normalization of an unusual notion of reduction on types, in which type variables may be replaced by their upper bounds from the context.

Our goal is to establish fundamental meta-theoretic results for $F^\omega_\leq$, leading up to sound and complete algorithms for checking the subtyping and typing relations. We begin in Section 2 by introducing $F^\omega_\leq$. Sections 3, 4, and 5 develop preliminary results needed in Section 6, the core of the paper, where the decidability of subtyping is proved. Section 7 extends the analysis to the decidability of typing.

In the technical development, we sometimes elide routine proofs. Those not shown are straightforward when performed in the order given.

2 Definition of $F^\omega_\leq$

Girard's System F^ω [Gir72] is a typed λ-calculus with higher-order polymorphism. Besides the term abstraction ($\mathit{fun}\,(x{:}T)\,t$) and application ($f\ a$) of the simply typed λ-calculus [Chu40] and the type abstraction ($\mathit{fun}\,(A{:}K)\,t$) and application ($t\ [T]$) of the second-order polymorphic λ-calculus [Gir72, Rey74], it includes the possibility of abstraction ($\mathit{Fun}\,(A{:}K)\,T$) and application ($T\ U$) within type expressions. To guarantee the well-formedness of applications within types, an extra level of *kinds* is introduced: the kind $\star$ classifies ordinary types (which are inhabited by terms), while kinds of the form $K_1{\rightarrow}K_2$ classify *type operators*: functions mapping types of kind K_1 to types of kind K_2. The basic typing judgement for $F^\omega_\leq$ is $\Gamma \vdash t \in T$, read "term t has type T in context Γ," where Γ records the type of each free term variable x and the kind of each free type variable A.

To extend F^ω with subtyping, we introduce an ordering $S \leq T$ on the elements of each kind K. The declaration of each type variable A in Γ is extended with an upper bound, written $A{\leq}T$, which constrains A to range only over subtypes of T in the appropriate kind. To allow new constraints of this form to be introduced

[2] Compagnoni [Com94] has independently achieved some results closely related to ours. After early joint work on the formulation of $F^\omega_\leq$ (c.f. [CP93]), vagaries of geography led our efforts onto separate, but parallel, tracks. Leaving aside inessential technical differences — our "reducing" system in Section 6.2 performs reduction in the premises of the rules, where her analogous "normalizing system" assumes that the conclusion is already normalized; she proves Church-Rosser by marking redices, while we adapt Tait and Martin-Löf's method of parallel reduction; etc. — the two proofs are broadly similar in structure. The major differences are as follows. 1) Compagnoni's results are for a more powerful system, $F^\omega_\wedge$, that includes intersection types in addition to the machinery of $F^\omega_\leq$. 2) Our development addresses the decidability of typechecking in addition to subtyping. 3) Compagnoni's version of the crucial substitution lemma (our Lemma 6.4.3) is phrased more cleverly and its proof requires a less intricate analysis. 4) Our proof of the termination of the subtyping algorithm is based on showing strong normalization for an extended notion of reduction in which type variables may be replaced by their upper bounds from the context; the argument is rather difficult, but introduces techniques that may be useful in a broader ranger of calculi including, for example, type abbreviations. Compagnoni uses a more direct term rewriting technique.

into the context, we extend the universal quantifier, $All(A{:}K)\,U$, to a *bounded quantifier* $All(A{\le}T)\,U$.[3]

To ensure that the new system can still type all the terms of F^{ω}, we assume that the subtype relation in every kind K has a maximal element $Top(K)$. The assumption $A{\le}Top(K)$ replaces $A{:}K$.

For kinds of the form $K_1{\rightarrow}K_2$, the subtype relation is just the pointwise extension of subtyping for K_2: a function $S \in K_1{\rightarrow}K_2$ is smaller than a function $T \in K_1{\rightarrow}K_2$ if $S\ U \le T\ U$ for every $U \in K_1$.[4]

At the base kind $\star$, the subtype relation also includes rules for the type constructors $T_1{\rightarrow}T_2$ and $All(A{\le}T_1)\,T_2$. The rule for arrow types embodies the familiar contravariant/covariant inclusion of function spaces:

$$\frac{\Gamma \vdash T_1 \le S_1 \qquad \Gamma \vdash S_2 \le T_2}{\Gamma \vdash S_1{\rightarrow}S_2 \le T_1{\rightarrow}T_2}$$

Intuitively, a function f whose results inhabit S_2 whenever its arguments inhabit S_1 may safely be substituted for a function in $T_1{\rightarrow}T_2$, provided that any element of T_1 that might be given as an argument to f can safely be used as an element of S_1 and that f's result, an element of S_2, can be used in place of the expected T_2.

The subtyping rule for bounded quantifiers is equally simple:

$$\frac{\Gamma, A{\le}U \vdash S_2 \le T_2}{\Gamma \vdash All(A{\le}U)\,S_2 \le All(A{\le}U)\,T_2}$$

That is, a polymorphic function $f \in All(A{\le}U)\,S_2$ can be used in a context that expects an element of $All(A{\le}U)\,T_2$, provided that, for each legal argument type T, the value of f at T can safely be used as an element of T_2.

It would be semantically sensible to refine the right-hand premise of this rule so that it only requires $S_2 \le T_2$ when A is constrained to the *common* part of their domain:

$$\frac{\Gamma \vdash T_1 \le S_1 \qquad \Gamma, A{\le}T_1 \vdash S_2 \le T_2}{\Gamma \vdash All(A{\le}S_1)\,S_2 \le All(A{\le}T_1)\,T_2}$$

This, indeed, is the form in which the rule appears in most presentations of second-order bounded quantification (c.f. [CP94] for a survey). However, the extra flexibility offered by this refinement does not seem to be useful in practice and it is very costly: this rule is responsible for the failure of a number of important proof-theoretic properties in standard formulations of $F_{\le}$ [Ghe93a, Ghe93b, GP92, Pie92], including decidability of subtyping.

Another variant of the quantifier subtyping rule allows the bounds to differ but requires that the bodies be in the subtype relation under the *trivial* assumption on the bound variable:

$$\frac{\Gamma \vdash T_1 \le S_1 \qquad \Gamma, A{\le}Top(K) \vdash S_2 \le T_2}{\Gamma \vdash All(A{\le}S_1)\,S_2 \le All(A{\le}T_1)\,T_2}$$

(where K is the kind of S_1 and T_1). Indeed, an earlier draft of this paper used this rule instead of the equal-bounds variant. All of the results about subtyping hold for both systems (with nearly identical proofs). But the algorithm for synthesizing minimal types works only for the equal-bounds rule, and not (as we erroneously claimed) for the top-rule. Indeed, the top-rule actually destroys the minimal typing property! For example, in the context

$$\Gamma = Y{\le}Top(\star)$$

the term

$$e = fun\,(X{\le}Y)\ \ fun\,(x{:}X)\ \ x$$

has both of the types

$$All(X{\le}Y)\,X \rightarrow X$$
$$All(X{\le}Y)\,X \rightarrow Y,$$

but these types are incomparable (using the top-rule) and have no common lower bound. We are endebted to Giorgio Ghelli for this example.

[3] We could also extend type operators $Fun\,(A{:}K)\,U$ to bounded operators $Fun\,(A{\le}T)\,U$, but it is not clear that this refinement would be useful in practice, and it would complicate the metatheory, since we would then need to introduce a sub*kinding* relation.

[4] Again, richer definitions of operator subtyping are possible: for example, we might allow monotone subtyping, antimonotone subtyping, etc. [Car90]. This extension *does* seem useful in practice (e.g. [HP94]), but its algorithmic implications are unclear.

2.1 Syntax

The kinds, types, terms, and contexts of $F^{\omega}_{\leq}$ are defined by the following abstract grammar:

K	$::=$	$\star$	kind of types
	$\mid$	$K \rightarrow K$	kind of type operators
T	$::=$	A	type variable
	$\mid$	$\mathit{Fun}\,(A{:}K)\,T$	type operator
	$\mid$	$T\ T$	application of a type operator
	$\mid$	$\mathit{Top}(K)$	maximal type
	$\mid$	$T \rightarrow T$	function type
	$\mid$	$\mathit{All}(A {\leq} T)\,T$	universally quantified type
t	$::=$	x	variable
	$\mid$	$\mathit{fun}\,(x{:}T)\,t$	abstraction
	$\mid$	$t\ t$	application
	$\mid$	$\mathit{fun}\,(A {\leq} T)\,t$	type abstraction
	$\mid$	$t\ T$	type application
Γ	$::=$	$\bullet$	empty context
	$\mid$	$\Gamma,\, x{:}T$	variable binding
	$\mid$	$\Gamma,\, A {\leq} T$	type variable binding with bound

The inference rules that follow define sets of derivable *statements* of the following forms:

$\vdash \Gamma\ \mathit{ok}$	Γ is a well-formed context
$\Gamma \vdash T \in K$	type T has kind K in context Γ
$\Gamma \vdash S \leq T$	S is a subtype of T in Γ
$\Gamma \vdash t \in T$	term t has type T in Γ.

Terms, types, contexts, and statements that differ only in the names of bound variables are regarded as identical.

2.2 Contexts and Kinding

Well-formed contexts are constructed from the empty context by adding well-kinded type and term variable declarations.

$$\vdash\ \bullet\ \mathit{ok} \qquad \text{(C-Empty)}$$

$$\frac{\Gamma\ \vdash\ T\ \in\ K \qquad A\ \notin\ \mathrm{dom}(\Gamma)}{\vdash\ \Gamma,\, A {\leq} T\ \ \mathit{ok}} \qquad \text{(C-TVar)}$$

$$\frac{\Gamma\ \vdash\ T\ \in\ \star \qquad x\ \notin\ \mathrm{dom}(\Gamma)}{\vdash\ \Gamma,\, x{:}T\ \ \mathit{ok}} \qquad \text{(C-Var)}$$

Since the side conditions guarantee that the bindings in well-formed contexts are always for distinct variables, we often consider them as finite functions from variables to types; for example, the upper bound of A in Γ is written $\Gamma(A)$. We write $\mathrm{dom}(\Gamma)$ for the set of term and type variables bound by Γ. If Γ is a prefix of Γ', we say that Γ' is an *extension* of Γ.

The definition of the kinding relation is standard. Type variables have the same kind as their upper bounds; abstraction and application provide introduction and elimination forms for arrow-kinds; $\mathit{Top}(K)$ has kind K; arrow- and All-types are well-kinded if their components are. We maintain the invariant that kinding statements are only derivable in well-formed contexts.

$$\frac{\Gamma \vdash \Gamma(A) \in K}{\Gamma \vdash A \in K} \qquad \text{(K-TVar)}$$

$$\frac{\Gamma, A \leq Top(K_1) \vdash T \in K_2}{\Gamma \vdash Fun(A{:}K_1)T \in K_1 \rightarrow K_2} \qquad \text{(K-Arrow-I)}$$

$$\frac{\Gamma \vdash S \in K_1 \rightarrow K_2 \qquad \Gamma \vdash T \in K_1}{\Gamma \vdash S\ T \in K_2} \qquad \text{(K-Arrow-E)}$$

$$\frac{\vdash \Gamma\ ok}{\Gamma \vdash Top(K) \in K} \qquad \text{(K-Top)}$$

$$\frac{\Gamma \vdash T_1 \in \star \qquad \Gamma \vdash T_2 \in \star}{\Gamma \vdash T_1 \rightarrow T_2 \in \star} \qquad \text{(K-Arrow)}$$

$$\frac{\Gamma, A \leq T_1 \vdash T_2 \in \star}{\Gamma \vdash All(A \leq T_1)T_2 \in \star} \qquad \text{(K-All)}$$

(The kinding and context well-formedness judgements are mutually recursive, but the two main judgement forms — subtyping and typing — only depend non-recursively on other judgements.)

2.3 Conversion

The presence of abstractions and applications in type expressions leads us to consider *conversion* within types. For technical convenience, we use a slight extension of the standard β-conversion relation: in addition to reductions of the usual form $(Fun(A{:}K_1)T)\ U \longrightarrow_{\beta} [U/A]T$ we allow reductions of the form $Top(K_1 \rightarrow K_2)\ T \longrightarrow_{\top} Top(K_2)$, which relate the maximal elements of different kinds. We could achieve the same effect by extending the rule S-Top below, but this way seems cleaner.

2.3.1 Definition [$\beta\top$-reduction]: One-step $\beta\top$-reduction is the smallest relation on types closed under the following rules:

$$\frac{}{Top(K_1 \rightarrow K_2)\ S \longrightarrow_{\beta\top} Top(K_2)}\ (\top) \qquad \frac{}{(Fun(A{:}K)S)\ T \longrightarrow_{\beta\top} [T/A]S}\ (\beta)$$

$$\frac{S \longrightarrow_{\beta\top} S'}{S\ T \longrightarrow_{\beta\top} S'\ T} \qquad \frac{T \longrightarrow_{\beta\top} T'}{S\ T \longrightarrow_{\beta\top} S\ T'}$$

$$\frac{S \longrightarrow_{\beta\top} S'}{(S \rightarrow T) \longrightarrow_{\beta\top} (S' \rightarrow T)} \qquad \frac{T \longrightarrow_{\beta\top} T'}{(S \rightarrow T) \longrightarrow_{\beta\top} (S \rightarrow T')}$$

$$\frac{S \longrightarrow_{\beta\top} S'}{All(A \leq S)T \longrightarrow_{\beta\top} All(A \leq S')T} \qquad \frac{T \longrightarrow_{\beta\top} T'}{All(A \leq S)T \longrightarrow_{\beta\top} All(A \leq S)T'}$$

$$\frac{S \longrightarrow_{\beta\top} S'}{Fun(A{:}K)S \longrightarrow_{\beta\top} Fun(A{:}K)S'}$$

The many-step $\beta\top$-reduction relation $\longrightarrow^{\star}_{\beta\top}$ is the reflexive and transitive closure of one-step reduction; $=_{\beta\top}$ is its reflexive, transitive, and symmetric closure. When T has a normal form (it will necessarily be unique), we denote it by $T^{!}$. Reduction to $\beta\top$-normal form is written $\longrightarrow^{!}_{\beta\top}$.

2.4 Subtyping

The $F^\omega_\leq$ subtyping relation $\Gamma \vdash S \leq T$ is a straightforward extension of the subtyping relation of $F_\leq$ [CW85, CG92, CP94]. We start by stipulating that $\beta\top$-convertible types always lie in the same equivalence class in the subtype ordering (S-Conv), and that the subtype relation at every kind is reflexive (R-Refl) and transitive (R-Trans). Type assumptions from the context may be used as axioms (S-TVar). $\mathit{Top}(K)$ is maximal in the ordering for kind K (S-Top). Type operators (S-Abs) and applications (S-App) are subtyped pointwise. Arrow- and All-types have the rules discussed above (S-Arrow and S-All).

In several places in the definition, we add premises to ensure that a proper kinding discipline is respected (e.g. $S \leq \mathit{Top}(K)$ only when $S \in K$, etc.). But for readability, these are kept to a minimum: we maintain the invariant that whenever the conclusion of a subtyping judgement is well-kinded, the types on the right- and left-hand sides of the $\leq$ will have the same kind and all of the subderivations will be similarly well behaved.

$$\frac{\Gamma \vdash S \leq U \qquad \Gamma \vdash U \in K \qquad U =_{\beta\top} T}{\Gamma \vdash S \leq T} \tag{S-Conv}$$

$$\Gamma \vdash T \leq T \tag{S-Refl}$$

$$\frac{\Gamma \vdash S \leq U \qquad \Gamma \vdash U \leq T \qquad \Gamma \vdash U \in K}{\Gamma \vdash S \leq T} \tag{S-Trans}$$

$$\Gamma \vdash A \leq \Gamma(A) \tag{S-TVar}$$

$$\frac{\Gamma \vdash S \in K}{\Gamma \vdash S \leq \mathit{Top}(K)} \tag{S-Top}$$

$$\frac{\Gamma, A {\leq} \mathit{Top}(K) \vdash S \leq T}{\Gamma \vdash \mathit{Fun}(A{:}K)\,S \leq \mathit{Fun}(A{:}K)\,T} \tag{S-Abs}$$

$$\frac{\Gamma \vdash S \leq T}{\Gamma \vdash S\,U \leq T\,U} \tag{S-App}$$

$$\frac{\Gamma \vdash T_1 \leq S_1 \qquad \Gamma \vdash S_2 \leq T_2}{\Gamma \vdash S_1{\rightarrow}S_2 \leq T_1{\rightarrow}T_2} \tag{S-Arrow}$$

$$\frac{\Gamma, A{\leq}U \vdash S_2 \leq T_2}{\Gamma \vdash \mathit{All}(A{\leq}U)\,S_2 \leq \mathit{All}(A{\leq}U)\,T_2} \tag{S-All}$$

2.5 Typing

The typing relation $\Gamma \vdash t \in T$ is standard [CW85, CG92], modulo a few extra kinding assumptions. The rule T-Subsumption captures the intended interpretation of subtyping as "safe substitutability." The others are straightforward extensions of the arrow- and All-introduction and -elimination rules of pure F^ω.

$$\frac{\Gamma \vdash s \in S \qquad \Gamma \vdash T \in \star \qquad \Gamma \vdash S \leq T}{\Gamma \vdash s \in T} \tag{T-Subsumption}$$

$$\frac{\vdash \Gamma\ \mathit{ok}}{\Gamma \vdash x \in \Gamma(x)} \tag{T-Var}$$

$$\frac{\Gamma, x{:}T_1 \vdash t \in T_2}{\Gamma \vdash \mathit{fun}(x{:}T_1)\,t \in T_1{\rightarrow}T_2} \tag{T-Arrow-I}$$

$$\frac{\Gamma \vdash f \in T_1{\rightarrow}T_2 \qquad \Gamma \vdash a \in T_1}{\Gamma \vdash f\,a \in T_2} \tag{T-Arrow-E}$$

$$\frac{\Gamma, A{\leq}T_1 \vdash t \in T_2}{\Gamma \vdash \mathit{fun}(A{\leq}T_1)\,t \in \mathit{All}(A{\leq}T_1)\,T_2} \tag{T-All-I}$$

$$\frac{\Gamma \vdash f \in \mathit{All}(A{\leq}T_1)\,T_2 \qquad \Gamma \vdash S \in K \qquad \Gamma \vdash S \leq T_1}{\Gamma \vdash f\,S \in [S/A]T_2} \tag{T-All-E}$$

3 Properties of Reduction

We now pause to establish some technical properties of the reduction relation and to define an auxiliary notion of *parallel reduction* that will simplify some of the inductive arguments in later sections. The main result of this section is the Church-Rosser property, by a straightforward adaptation of Tait and Martin-Löf's proof for ordinary β-reduction (c.f. [Bar84]).

3.1 Definition [Parallel reduction]: Single-step parallel reduction is the least relation closed under the following rules:

$$\frac{}{\mathit{Top}(K_1 \rightarrow K_2)\ T \twoheadrightarrow_{\beta\top} \mathit{Top}(K_2)} \qquad \frac{S \twoheadrightarrow_{\beta\top} S' \qquad T \twoheadrightarrow_{\beta\top} T'}{(\mathit{Fun}(A{:}K)\ S)\ T \twoheadrightarrow_{\beta\top} [T'/A]S'}$$

$$\frac{S \twoheadrightarrow_{\beta\top} S' \qquad T \twoheadrightarrow_{\beta\top} T'}{S\ T \twoheadrightarrow_{\beta\top} S'\ T'} \qquad \frac{T \twoheadrightarrow_{\beta\top} T'}{\mathit{Fun}(A{:}K)\ T \twoheadrightarrow_{\beta\top} \mathit{Fun}(A{:}K)\ T'}$$

$$\frac{S \twoheadrightarrow_{\beta\top} S' \qquad T \twoheadrightarrow_{\beta\top} T'}{S \rightarrow T \twoheadrightarrow_{\beta\top} S' \rightarrow T'} \qquad \frac{S \twoheadrightarrow_{\beta\top} S' \qquad T \twoheadrightarrow_{\beta\top} T'}{\mathit{All}(A{\leq}S)\ T \twoheadrightarrow_{\beta\top} \mathit{All}(A{\leq}S')\ T'}$$

$$\frac{}{T \twoheadrightarrow_{\beta\top} T}$$

Ordinary single-step reduction is a subrelation of single-step parallel reduction, which is a subrelation of multi-step ordinary reduction. The reflexive, transitive closures of the two relations coincide:

3.2 Fact:

1. $\longrightarrow_{\beta\top} \quad \subset \quad \twoheadrightarrow_{\beta\top}$
2. $\longrightarrow^{\star}_{\beta\top} \quad \supset \quad \twoheadrightarrow_{\beta\top}$
3. $\longrightarrow^{\star}_{\beta\top} \quad = \quad \twoheadrightarrow^{\star}_{\beta\top}$

Substitution commutes with parallel and multi-step reduction:

3.3 Lemma:

1. If $S \twoheadrightarrow_{\beta\top} S'$ and $T \twoheadrightarrow_{\beta\top} T'$ then $[T/A]S \twoheadrightarrow_{\beta\top} [T'/A]S'$.
2. If $S \twoheadrightarrow^{\star}_{\beta\top} S'$ and $T \twoheadrightarrow^{\star}_{\beta\top} T'$ then $[T/A]S \twoheadrightarrow^{\star}_{\beta\top} [T'/A]S'$.
3. If $S \longrightarrow^{\star}_{\beta\top} S'$ and $T \longrightarrow^{\star}_{\beta\top} T'$ then $[T/A]S \longrightarrow^{\star}_{\beta\top} [T'/A]S'$.

In the proof of 3.3, we need the following property of substitution:

3.4 Fact: If $A \neq A'$ and $A' \notin FV(S)$, then $[S/A]([T/A']U) = [[S/A]T/A']([S/A]U)$.

One useful consequence of 3.3(3) is that if an expression with an outermost redex has a reduction path in which this redex is reduced at some point, then this reduction can be performed first without changing the result:

3.5 Corollary [Outermost reduction]: If $(\mathit{Fun}(A{:}K)S)T \longrightarrow^{\star}_{\beta\top} U$, where $U \neq (\mathit{Fun}(A{:}K)\,S')T'$ with $S \longrightarrow^{\star}_{\beta\top} S'$, and $T \longrightarrow^{\star}_{\beta\top} T'$, then $[T/A]S \longrightarrow^{\star}_{\beta\top} U$.

3.6 Lemma [Diamond property for $\twoheadrightarrow_{\beta\top}$]: For all types S, S_1, and S_2 with $S \twoheadrightarrow_{\beta\top} S_1$ and $S \twoheadrightarrow_{\beta\top} S_2$, there is a type S_3 such that $S_1 \twoheadrightarrow_{\beta\top} S_3$ and $S_2 \twoheadrightarrow_{\beta\top} S_3$.

$$
\begin{tikzcd}
 & S \arrow[dl, twoheadrightarrow, "1"', "\beta\top"] \arrow[dr, twoheadrightarrow, "1", "\beta\top"'] & \\
S_1 \arrow[dr, dotted, twoheadrightarrow, "1", "\beta\top"'] & & S_2 \arrow[dl, dotted, twoheadrightarrow, "1"', "\beta\top"] \\
 & S_3 &
\end{tikzcd}
$$

Proof: Straightforward extension of the standard argument (c.f. [Bar84]). □

3.7 Corollary [Church-Rosser for $\longrightarrow^{\star}_{\beta\top}$]: For all types S, S_1, and S_2 with $S \longrightarrow^{\star}_{\beta\top} S_1$ and $S \longrightarrow^{\star}_{\beta}$ S_2, there is a type S_3 such that $S_1 \longrightarrow^{\star}_{\beta\top} S_3$ and $S_2 \longrightarrow^{\star}_{\beta\top} S_3$.

The proof of strong normalization for $\longrightarrow_{\beta\top}$ has to be deferred until after we have studied the properties of the kinding system, since our proof of normalization requires that the types involved by well kinded.

One more property of $\twoheadrightarrow_{\beta\top}$ will be needed for the induction in the proof of Lemma 6.4.8.

3.8 Lemma [$\twoheadrightarrow_{\beta\top}$ and $\longrightarrow^{\star}_{\beta\top}$]: If $S \twoheadrightarrow_{\beta\top} S_1$ and $S \longrightarrow^{\star}_{\beta\top} S_2$, then there is an S_3 with $S_1 \longrightarrow^{\star}_{\beta\top} S_3$ and $S_2 \twoheadrightarrow_{\beta\top} S_3$.

$$
\begin{tikzcd}
 & S \arrow[dl, twoheadrightarrow, "\beta\top"] \arrow[dr, "\star", "\beta\top"'] & \\
S_1 \arrow[dr, dotted, "\star", "\beta\top"'] & & S_2 \arrow[dl, dotted, twoheadrightarrow, "\beta\top"] \\
 & S_3 &
\end{tikzcd}
$$

Proof: By induction on the length of $S \longrightarrow^{\star}_{\beta\top} S_2$, using 3.2(1) and Church-Rosser. □

4 Kinding

Next, we state some basic technical properties of the context well-formedness and kinding judgements.

4.1 Lemma [Generation of contexts]:

1. If $\vdash \Gamma$ *ok*, then:
   (a) $\Gamma = \bullet$; or
   (b) $\Gamma = \Gamma_1, x{:}T$, with $\vdash \Gamma_1$ *ok* and $\Gamma_1 \vdash T \in \star$ as subderivations; or
   (c) $\Gamma = \Gamma_1, A{\leq}T$, with $\vdash \Gamma_1$ *ok* and $\Gamma_1 \vdash T \in K$ for some K as subderivations.
2. If $\Gamma \vdash S \in K$, then $\vdash \Gamma$ *ok* as a subderivation.

4.2 Lemma [Generation of types]:

1. If $\Gamma \vdash A \in K$, then $\Gamma \vdash \Gamma(A) \in K$.
2. If $\Gamma \vdash \mathit{Fun}(A{:}K_1)T \in K$, then, for some K_2, we have $\Gamma, A {\leq} \mathit{Top}(K_1) \vdash T \in K_2$ and $K = K_1 {\rightarrow} K_2$.
3. If $\Gamma \vdash S\ T \in K$, then, for some K', we have $\Gamma \vdash S \in K' {\rightarrow} K$ and $\Gamma \vdash T \in K'$.
4. If $\Gamma \vdash S {\rightarrow} T \in K$, then $K = \star$ and $\Gamma \vdash S, T \in \star$.
5. If $\Gamma \vdash \mathit{Top}(K) \in K'$, then $K = K'$.

6. If $\Gamma \vdash All(A{\leq}S)T \in K$, then $K = \star$ and $\Gamma, A{\leq}S \vdash T \in \star$.

Moreover, the implied derivations are all subderivations of the originals.

We prove the decidability of the kinding system by showing that it is equivalent to a different system whose decidability is obvious.

4.3 Definition [Algorithmic kinding]: The algorithmic kinding relation $\Gamma \vdash_{\mathcal{A}} T \in K$ is the least relation closed under the kinding rules, where instead of rule K-TVAR we use the following:

$$\frac{\Gamma_1 \vdash_{\mathcal{A}} T \in K \qquad \vdash_{\mathcal{A}} \Gamma_1, A{\leq}T, \Gamma_2 \;\; ok}{\Gamma_1, A{\leq}T, \Gamma_2 \vdash_{\mathcal{A}} A \in K} \tag{K-TVAR$'$}$$

The algorithmic context well-formedness relation $\vdash_{\mathcal{A}} \Gamma \;\; ok$ is defined as before, using algorithmic kinding.

4.4 Lemma [Context strengthening for algorithmic kinding]:

1. If $\Gamma_1, A{\leq}S, \Gamma_2 \vdash_{\mathcal{A}} T \in K$ and A is not free in Γ_2 or in T, then $\Gamma_1, \Gamma_2 \vdash_{\mathcal{A}} T \in K$.

2. If $\vdash_{\mathcal{A}} \Gamma_1, A{\leq}S, \Gamma_2 \;\; ok$ and A is not free in Γ_2, then $\vdash_{\mathcal{A}} \Gamma_1, \Gamma_2 \;\; ok$.

3. If $\Gamma_1, x{:}S, \Gamma_2 \vdash_{\mathcal{A}} T \in K$, then $\Gamma_1, \Gamma_2 \vdash_{\mathcal{A}} T \in K$.

4. If $\vdash_{\mathcal{A}} \Gamma_1, x{:}S, \Gamma_2 \;\; ok$, then $\vdash_{\mathcal{A}} \Gamma_1, \Gamma_2 \;\; ok$.

4.5 Lemma [Decidability of kinding]: The relations $\vdash \Gamma \;\; ok$ and $\Gamma \vdash S \in K$ are decidable.

Proof: It is easy to prove by induction that the two kinding systems and the two definitions of context well-formedness are equivalent. In each direction, we only have to consider the rule for variables, since all other rules coincide.

Case K-TVAR: $\Gamma \vdash \Gamma(A) \in K$

By Lemma 4.1(2) $\vdash \Gamma_1, A{\leq}T, \Gamma_2 \;\; ok$ as subderivation. So by the induction hypothesis, $\vdash_{\mathcal{A}} \Gamma_1, A{\leq}T, \Gamma_2 \;\; ok$ and $\Gamma_1, A{\leq}T, \Gamma_2 \vdash_{\mathcal{A}} T \in K$. Repeated application of Lemma 4.4 yields $\Gamma_1 \vdash_{\mathcal{A}} T \in K$.

Case K-TVAR$'$: $\Gamma_1 \vdash_{\mathcal{A}} T \in K$ and $\Gamma \vdash_{\mathcal{A}} \Gamma_1, A{\leq}T, \Gamma_2 \;\; ok$

By the induction hypothesis, $\Gamma_1 \vdash T \in K$ and $\vdash \Gamma_1, A{\leq}T, \Gamma_2 \;\; ok$, so the result follows by K-TVAR and weakening.

Now, the algorithm obtained by reading the algorithmic kinding rules from bottom to top as Horn clauses always terminates, since in each step the total number of characters in the conclusion is greater than the number of characters in any of the premises. Since the systems are equivalent, $\Gamma \vdash S{:}K$ is also decidable. $\Box$

4.6 Lemma [Uniqueness of kinding]: If $\Gamma \vdash S \in K$ and $\Gamma \vdash S \in K'$, then $K = K'$.

This justifies the following notation:

4.7 Definition: The unique kind of a well-kinded type S in a context Γ is written $Kind_\Gamma(S)$.

4.8 Lemma [Transposition and weakening for kinding]: Suppose that Γ' is a well-formed extension of $\Gamma_1, A'{\leq}T, A{\leq}S, \Gamma_2$. If $\Gamma_1, A{\leq}S, A'{\leq}T, \Gamma_2 \vdash U \in K$ and $A \notin FV(T)$, then $\Gamma' \vdash U \in K$.

4.9 Lemma [Context update for kinding]: If $\Gamma_1, A{\leq}S, \Gamma_2 \vdash T \in K$ and $\Gamma_1 \vdash S, S' \in K'$, then $\Gamma_1, A{\leq}S', \Gamma_2 \vdash T \in K$.

4.10 Lemma [Top reduction]: If $\Gamma \vdash Top(K)\; T_1 \ldots T_n \in K'$, then $Top(K)\; T_1 \ldots T_n \longrightarrow^{*}_{\beta\top} Top(K')$.

4.11 Lemma [Kinding and substitution]: Suppose $\Gamma_1 \vdash T \in K'$.

1. If $\vdash \Gamma_1, A{\leq}Top(K'), \Gamma_2 \;\; ok$, then $\vdash \Gamma_1, [T/A]\Gamma_2 \;\; ok$.

2. If $\Gamma_1, A{\leq}Top(K'), \Gamma_2 \vdash S \in K$, then $\Gamma_1, [T/A]\Gamma_2 \vdash [T/A]S \in K$.

Proof: Both parts are proved simultaneously by induction on derivations. □

4.12 Lemma [Subject reduction for types and contexts]:

1. If $\Gamma \vdash S \in K$ and $S \longrightarrow^{\star}_{\beta\top} T$ then $\Gamma \vdash T \in K$.
2. If $\Gamma \vdash S \in K$ and $\Gamma \longrightarrow^{\star}_{\beta\top} \Gamma'$, then $\Gamma' \vdash S \in K$.

4.13 Corollary [Kind invariance under conversion]: If $S =_{\beta\top} T$, where $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$, then $K_S = K_T$.

5 Strong Normalization of Types

We shall often need the fact that $\beta\top$-reduction is strongly normalizing for well-kinded types. In fact, we prove the strong normalization of a more general reduction relation, called $\beta\top\Gamma$-reduction, which will be used to prove the termination of the subtyping and typing algorithms. Besides the usual β and $\top$ reductions, we allow variables to be replaced by their upper bounds from the context. Γ-reduction is reminiscent of the common operation in type checker and proof checker implementations of replacing a type definition by its expansion (c.f. [SP93]).

We begin by proving the strong normalization of $\beta\top$-reduction, using by a straightforward translation argument.

5.1 Definition: Define a family of types T_K, one for each kind K, as follows:

$$\begin{array}{lcl} T_\star & = & \mathit{All}(A{:}\star)\, A \\ T_{K_1 \to K_2} & = & \mathit{Fun}(A{:}K_1)\ T_{K_2} \end{array}$$

Note that $\bullet \vdash T_K \in K$ for each K.

5.2 Lemma [Strong $\beta\top$-normalization]: Suppose $\Gamma \vdash S \in K$. Then there is no infinite $\beta\top$-reduction from S.

Proof: Define a translation function F mapping $F^\omega_\leq$ types to F^ω types:

$$\begin{array}{lcl} F(\mathit{Top}(K)) & = & T_K \\ F(\mathit{All}(A{\leq}S)\, T) & = & \mathit{All}(A)\, F(S) \to F(T) \\[4pt] F(\mathit{Fun}(A{:}K)\, S) & = & \mathit{Fun}(A{:}K)\ F(S) \\ F(S\ T) & = & F(S)\ F(T) \\ F(S \to T) & = & F(S) \to F(T). \end{array}$$

On contexts, F replaces each type variable binding $A{\leq}T$ in Γ by the kinding assumption $A{:}K$, where K is the kind of T in Γ. It is easy to check that if $\Gamma \vdash S \in K$ in $F^\omega_\leq$, then $F(\Gamma) \vdash F(S) \in K$ in F^ω.

Now, any $\beta\top$-reduction in $F^\omega_\leq$ from S can be mirrored by a β-reduction from $F(S)$ of the same length in F^ω. The existence of an infinite $\beta\top$-reduction in $F^\omega_\leq$ would thus contradict the strong normalization of F^ω [Gir72, Gal90]. □

Next, we define the notion of Γ-reduction and establish some of its basic properties.

5.3 Definition: Single-step Γ-*reduction* is the least family of relations closed under:

$$\frac{\Gamma(A) \neq \mathit{Top}(K)}{A \longrightarrow_\Gamma \Gamma(A)}\ (\Gamma) \qquad \frac{S \longrightarrow_{(\Gamma,\ A\leq \mathit{Top}(K))} S'}{\mathit{Fun}(A{:}K)S \longrightarrow_\Gamma \mathit{Fun}(A{:}K)S'}$$

$$\frac{S \longrightarrow_\Gamma S'}{S\ T \longrightarrow_\Gamma S'\ T} \qquad \frac{T \longrightarrow_\Gamma T'}{S\ T \longrightarrow_\Gamma S\ T'}$$

$$\frac{S \longrightarrow_\Gamma S'}{(S \rightarrow T) \longrightarrow_\Gamma (S' \rightarrow T)} \qquad \frac{T \longrightarrow_\Gamma T'}{(S \rightarrow T) \longrightarrow_\Gamma (S \rightarrow T')}$$

$$\frac{S \longrightarrow_\Gamma S'}{\mathit{All}(A{\leq}S)T \longrightarrow_\Gamma \mathit{All}(A{\leq}S')T} \qquad \frac{T \longrightarrow_{(\Gamma,\ A\leq S)} T'}{\mathit{All}(A{\leq}S)T \longrightarrow_\Gamma \mathit{All}(A{\leq}S)T'}$$

Single-step $\beta\top\Gamma$-reduction, written $\longrightarrow_{\beta\top\Gamma}$, is the least family of relations closed under these rules and the rules (β) and ($\top$) of Definition 2.3.1. The corresponding multi-step reductions are defined as usual. Note that in a multi-step Γ-reduction sequence, the Γ at each stage remains the same; Γ is only extended "internally," in the course of a single reduction, to keep track of variable bindings in those rules that define reduction under binders.

Note that we are careful to separate *All*-bound variables, whose bounds may be different from *Top* and which may thus act as Γ-redexes, from *Fun*-bound variables, whose bounds are always *Top* and which can never be Γ-reduced.

5.4 Lemma [Strong Γ-normalization]: If $\Gamma \vdash S \in K$, then there is no infinite Γ-reduction from S.

Proof: We show, by induction on the defintion of $\longrightarrow_\Gamma$, that if $S \longrightarrow_\Gamma T$ in one step, then $\Gamma \vdash T \in K$ by a shorter derivation. Most cases are straightforward; we list only the the ones for type variables and for *All*-types.

Case: $S = A$ and $T = \Gamma(A)$
By the generation lemma for types, $\Gamma \vdash \Gamma(A) \in K$ by a shorter derivation.

Case: $S = \mathit{All}(A{\leq}S_1)S_2$ and $T = \mathit{All}(A{\leq}S_1')S_2$
By the generation lemma for types and contexts and Lemma 4.1, $\Gamma \vdash S_1 \in K_1$, so by the induction hypothesis $\Gamma \vdash S_1' \in K_1$ by a shorter derivation. Finally $\Gamma \vdash \mathit{All}(A{\leq}S_1')S_2 \in \star$ by K-All, by a shorter derivation than the original.

Case: $S = \mathit{All}(A{\leq}S_1)S_2$ and $T = \mathit{All}(A{\leq}S_1)S_2'$
By the generation lemma $\Gamma,\ A{\leq}S_1 \vdash S_2 \in K_2$. The induction hypothesis gives $\Gamma,\ A{\leq}S_1 \vdash S_2' \in K_2$ by a shorter derivation, so by K-All we have $\Gamma \vdash \mathit{All}(A{\leq}S_1)S_2'$ by a shorter derivation than the original. □

We shall often use the subject-reduction property silently in what follows, to guarantee that a reduction sequence from a well-kinded term only contains well-kinded terms.

5.5 Lemma [Subject reduction]: If $\Gamma \vdash S \in K$ and $S \longrightarrow^\star_{\beta\top\Gamma} S'$, then $\Gamma \vdash S' \in K$.

Proof: By induction on the length of the reduction $S \longrightarrow^\star_{\beta\top\Gamma} S'$, with an inner induction on the definition of single-step $\beta\top\Gamma$-reduction. □

5.6 Fact:

1. If $\mathit{All}(A{\leq}T_1)\,T_2 \longrightarrow_\Gamma{}^\star \mathit{All}(A{\leq}V_1)\,V_2$, then $T_1 \longrightarrow_\Gamma{}^\star V_1$ and $T_2 \longrightarrow^\star_{\Gamma,\ A\leq T_1} V_2$.
2. If $\mathit{Fun}\,(A{:}K)\,T_2 \longrightarrow_\Gamma{}^\star \mathit{Fun}\,(A{:}K)\,V_2$, then $T_2 \longrightarrow^\star_{\Gamma,\ A\leq \mathit{Top}(K)} V_2$.
3. If $T_1\ T_2 \longrightarrow_\Gamma{}^\star V_1\ V_2$, then $T_1 \longrightarrow_\Gamma{}^\star V_1$ and $T_2 \longrightarrow_\Gamma{}^\star V_2$.

4. If $(T_1 \to T_2) \longrightarrow_{\Gamma}^{\star} (V_1 \to V_2)$, then $T_1 \longrightarrow_{\Gamma}^{\star} V_1$ and $T_2 \longrightarrow_{\Gamma}^{\star} V_2$.

Proof: We give the proof in detail for part (1); the rest are similar, but simpler. For part (1), we prove the more refined statement

> if $T_1 \longrightarrow_{\Gamma}^{\star} U_1$ and $T_2 \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} U_2$ and $All(A{\leq}U_1)\, U_2 \xrightarrow{\alpha}{}_{\Gamma}^{\star} All(A{\leq}V_1)\, V_2$, then $T_1 \longrightarrow_{\Gamma}^{\star} V_1$ and $T_2 \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} V_2$,

by induction on the length of α.

Case: α is empty
Immediate.

Case: $All(A{\leq}U_1)\, U_2 \xrightarrow{\alpha_1}{}_{\Gamma} All(A{\leq}U_1)\, U_2' \xrightarrow{\alpha_2}{}_{\Gamma}^{\star} All(A{\leq}V_1)\, V_2$
(I.e., α consists of a single-step reduction α_1 followed by a multi-step reduction α_2, where α_1 replaces a single variable in U_2 by its upper bound to yield U_2'.) To apply the induction hypothesis, we need to check that $U_2 \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} U_2'$, which immediately gives $T_2 \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} U_2'$.

But if α_1 replaces an occurrence of A by U_1 in U_2, i.e. $U_2 = U_2[A] \longrightarrow_{(\Gamma,\, A \leq U_1)} U_2[U_1]$, then we can build a reduction $U_2[A] \longrightarrow_{(\Gamma,\, A \leq T_1)} U_2[T_1] \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} U_2[U_1]$ by replacing this occurrence of A with T_1 and then using the assumption that $T_1 \longrightarrow_{\Gamma}^{\star} U_1$ (and hence $T_1 \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} U_1$) to develop T_1 to U_1 in-place. On the other hand, if α_1 replaces some other variable, then $U_2 \longrightarrow_{(\Gamma,\, A \leq T_1)}^{\star} U_2'$ is immediate. In both cases, the induction hypothesis then applies, directly yielding the desired result.

Case: $All(A{\leq}U_1)\, U_2 \xrightarrow{\alpha_1}{}_{\Gamma} All(A{\leq}U_1')\, U_2 \xrightarrow{\alpha_2}{}_{\Gamma}^{\star} All(A{\leq}V_1)\, V_2$
Here
the induction hypothesis applies directly (since $T_1 \longrightarrow_{\Gamma}^{\star} U_1 \longrightarrow_{\Gamma} U_1'$ and we have $T_2 \longrightarrow_{(\Gamma,\, A \leq T1)}^{\star} U_2$ by assumption) to yield the desired result. □

5.7 Lemma [Weak diamond property for Γ-reduction]:

$$
\begin{tikzcd}
T \arrow[r, "\Gamma"] \arrow[d, "\Gamma"'] & U \arrow[d, dotted, "\star\,\Gamma"] \\
V \arrow[r, dotted, "\star", "\Gamma"'] & W.
\end{tikzcd}
$$

Proof: By induction on the form of T.

Case: $T = A$
Then $V = U = \Gamma(A)$ and we may take $W = \Gamma(A)$.

Case: $T = All(A{\leq}T_1)\, T_2$
We must find $W = All(A{\leq}W_1)\, W_2$ such that the required diagram commutes; this will follow from the commutativity of a smaller diagram for T_1, U_1, V_1 and W_1 and another diagram for T_2, U_2, V_2 and W_2. There are three subcases to consider, depending on whether both of the reductions from T to U and V are in T_1, both are in T_2, or one is in T_1 and one in T_2. (Since the last case is symmetric, we may assume without loss of generality that T_1 is reduced to produce V and T_2 to produce U.)

Subcase: $U = All(A{\leq}U_1)\, T_2$ and $V = All(A{\leq}V_1)\, T_2$
Begin by applying the induction hypothesis to T_1, U_1, and V_1 to yield a common reduct W_1. We must then

show:

$$
\begin{tikzcd}
T_1 \arrow[rr, "\Gamma"] \arrow[dd, "\Gamma"'] & & U_1 \arrow[dd, "\Gamma"', "\star"] & \\
& T_2 \arrow[rr, "{\Gamma,\ A\leq T_1}", "="'] \arrow[dd, "{\Gamma,\ A\leq T_1}"', "="] & & T_2 \arrow[dd, dotted, "\star"', "{\Gamma,\ A\leq U_1}"] \\
V_1 \arrow[rr, "\Gamma", "\star"'] & & W_1 & \\
& T_2 \arrow[rr, dotted, "\star", "{\Gamma,\ A\leq V_1}"'] & & W_2.
\end{tikzcd}
$$

Set $W_2 = T_2$ and we are done.

Subcase: $U = All(A{\leq}T_1)\,U_2$ and $V = All(A{\leq}T_1)\,V_2$

We must find a W_2 such that

$$
\begin{tikzcd}
T_1 \arrow[rr, "\Gamma", "="'] \arrow[dd, "\Gamma"', "="] & & T_1 \arrow[dd, "\Gamma"', "="] & \\
& T_2 \arrow[rr, "{\Gamma,\ A\leq T_1}"] \arrow[dd, "{\Gamma,\ A\leq T_1}"'] & & U_2 \arrow[dd, dotted, "\star"', "{\Gamma,\ A\leq T_1}"] \\
T_1 \arrow[rr, "\Gamma", "="'] & & T_1 & \\
& V_2 \arrow[rr, dotted, "\star", "{\Gamma,\ A\leq T_1}"'] & & W_2.
\end{tikzcd}
$$

The existence of such a W_2 is given by the induction hypothesis.

Subcase: $U = All(A{\leq}T_1)\,U_2$ and $V = All(A{\leq}V_1)\,T_2$

Set $W_1 = V_1$. Then we must show:

$$
\begin{tikzcd}
T_1 \arrow[rr, "\Gamma", "="'] \arrow[dd, "\Gamma"'] & & T_1 \arrow[dd, "\Gamma"'] & \\
& T_2 \arrow[rr, "{\Gamma,\ A\leq T_1}"] \arrow[dd, "{\Gamma,\ A\leq T_1}"', "="] & & U_2 \arrow[dd, dotted, "\star"', "{\Gamma,\ A\leq T_1}"] \\
V_1 \arrow[rr, "\Gamma", "="'] & & V_1 & \\
& T_2 \arrow[rr, dotted, "\star", "{\Gamma,\ A\leq V_1}"'] & & W_2.
\end{tikzcd}
$$

If $T_2 \longrightarrow_{(\Gamma,\ A\leq T_1)} U_2$ by a Γ-reduction on some occurrence of A in T_2, then we have $T_2 = T_2[A]$ and $U_2 = T_2[T_1]$; set $W_2 = T_2[V_1]$. If $T_2 \longrightarrow_{(\Gamma,\ A\leq T_1)} U_2$ by a Γ-reduction on some occurrence of a variable other than A in T_2, then we can set $W_2 = U_2$, since $T_2 \longrightarrow^{\star}_{(\Gamma,\ A\leq V_1)} U_2$ follows directly from $T_2 \longrightarrow^{\star}_{(\Gamma,\ A\leq T_1)} U_2$ in this case.

Other cases:
Straightforward. □

5.8 Lemma [Church-Rosser for Γ-reduction]:

$$\begin{tikzcd} T \arrow[r, "\star", "\Gamma"'] \arrow[d, "\Gamma"', "\star"] & U \arrow[d, dotted, "\Gamma"', "\star"] \\ V \arrow[r, dotted, "\star", "\Gamma"'] & W. \end{tikzcd}$$

Proof: By Newman's Lemma, which states that the weak diamond property and strong normalization together imply Church-Rosser (c.f. [Bar84]). □

5.9 Lemma [Substitution commutes with Γ-reduction]: If $U \longrightarrow_{\Gamma}{}^{\star} V$ and $S \longrightarrow^{\star}_{(\Gamma,\ A \leq Top(K))} T$, then $[U/A]S \longrightarrow_{\Gamma}{}^{\star} [V/A]T$.

Proof: Since A's bound is $Top(K)$, it is not a $(\Gamma,\ A \leq Top(K))$-redex, so $[U/A]S \longrightarrow_{\Gamma}{}^{\star} [U/A]T$. Then $[U/A]T \longrightarrow_{\Gamma}{}^{\star} [V/A]T$ by applying the reduction from U to V at each point in $[U/A]T$ where A appeared in T. □

At this point, we can start proving properties relating Γ-reduction and $\beta\top$-reduction. First, a technical property that handles a key step of the following lemma.

5.10 Lemma: If $T_1 \longrightarrow^{\star}_{\beta\top} U_1$ and $T_2 \longrightarrow^{\star}_{(\Gamma,\ A \leq T_1)} V_2$, then there is some W_2 such that:

$$\begin{tikzcd} T_2 \arrow[d, "{\Gamma,\ A \leq T_1}"', "\star"] \arrow[dr, dotted, "{\Gamma,\ A \leq U_1}", "\star"'] & \\ V_2 \arrow[r, dotted, "\star", "\beta\top"'] & W_2. \end{tikzcd}$$

Proof: By induction on the length of the reduction from T_2 to V_2.

Case: $T_2 = V_2$
Then set $W_2 = T_2$ and we are done.

Case: $T_2 \longrightarrow^{\star}_{(\Gamma,\ A \leq T_1)} V_2' \longrightarrow_{(\Gamma,\ A \leq T_1)} V_2$
Apply the induction hypothesis to find a W_2' satisfying the desired property. We must now show:

$$\begin{tikzcd} T_2 \arrow[d, "{\Gamma,\ A \leq T_1}"', "\star"] \arrow[dr, "{\Gamma,\ A \leq U_1}", "\star"'] & \\ V_2' \arrow[r, "\star", "\beta\top"'] \arrow[d, "{\Gamma,\ A \leq T_1}"'] & W_2' \arrow[d, dotted, "\star"', "{\Gamma,\ A \leq U_1}"] \\ V_2 \arrow[r, dotted, "\star", "\beta\top"'] & W_2. \end{tikzcd}$$

If $V_2' \longrightarrow_{(\Gamma,\, A\leq T_1)} V_2$ by contracting a redex other than A, then $V_2' = V_2'[B]$ and $V_2 = V_2'[\Gamma(B)]$. In reducing from V_2' to W_2', this redex may be copied a number of times: $W_2' = W_2'[B][B]\ldots[B]$. Let W_2 be the result of contracting the residuals of this redex in W_2', i.e. $W_2 = W_2'[\Gamma(B)][\Gamma(B)]\ldots[\Gamma(B)]$. Similarly, if $V_2' = V_2'[A]$ and $V_2 = V_2'[T_1]$; again, let W_2 be the result of contracting the residuals of this redex in W_2'. Reduce $V_2 = V_2'[T_1]$ to $W_2 = W_2'[U_1][U_1]\ldots[U_1]$ by $V_2'[T_1] \longrightarrow^{\star}_{\beta\top} W_2'[T_1][T_1]\ldots[T_1] \longrightarrow^{\star}_{\beta\top} W_2'[U_1][U_1]\ldots[U_1]$. □

The next lemma establishes a confluence property for Γ and $\beta\top$ reductions. The proof is similar to that of Lemma 5.7. This lemma and Lemma 5.8 jointly handle the crucial step in the strong normalization argument that follows.

5.11 Lemma [$\beta\top$-reduction and Γ-reduction]:

$$\begin{tikzcd}
T \arrow[r, "\alpha", "\beta\top"'] \arrow[d, "\Gamma"', "\star"] & U \arrow[d, dotted, "\star"', "\Gamma"] \\
V \arrow[r, dotted, "\alpha'", "\beta\top\star"'] & W.
\end{tikzcd}$$

Moreover, α' has length at least 1.

Proof: By induction on α.

Case: $T = \mathit{Top}(K_1 \to K_2)\ T_1 \longrightarrow_{\top} \mathit{Top}(K_2) = U$

Any Γ-reduction from T must reduce only redexes in T_1, so V has the form $\mathit{Top}(K_1 \to K_2)\ V_1$. But then $V \longrightarrow_{\top} U$, and we can take $W = U$. Note that $V \longrightarrow_{\beta\top} W$ by a nonempty reduction.

Case: $T = (\mathit{Fun}(A{:}K)\ T_1)\ T_2 \longrightarrow_{\beta} [T_2/A]T_1 = U$

Any Γ-reduction from T consists of a number of separate reductions in T_1 and T_2 by Fact 5.6(2,3), so V has the form $(\mathit{Fun}(A{:}K)\ V_1)\ V_2$, with $T_1 \longrightarrow^{\star}_{(\Gamma,\, A\leq \mathit{Top}(K))} V_1$ and $T_2 \longrightarrow_{\Gamma}{}^{\star} V_2$. By Lemma 5.9, $[T_2/A]T_1 \longrightarrow_{\Gamma}{}^{\star} [V_2/A]V_1$, so we can take $W = [V_2/A]V_1$. Note that $V \longrightarrow_{\beta\top} W$ by a nonempty reduction.

Case:
$$\frac{T_1 \longrightarrow_{\beta\top} U_1}{T = \mathit{All}(A{\leq}T_1)\,T_2 \longrightarrow_{\beta\top} \mathit{All}(A{\leq}U_1)\,T_2 = U}$$

By Fact 5.6(1), V has the form $\mathit{All}(A{\leq}V_1)\,V_2$, with $T_1 \longrightarrow_{\Gamma}{}^{\star} V_1$ and $T_2 \longrightarrow^{\star}_{(\Gamma,\, A\leq T_1)} V_2$. Apply the induction hypothesis to find a W_1 with

$$\begin{tikzcd}
T_1 \arrow[r, "\beta\top"'] \arrow[d, "\Gamma"', "\star"] & T_1' \arrow[d, "\star"', "\Gamma"] \\
V_1 \arrow[r, "\alpha_1", "\beta\top\star"'] & W_1,
\end{tikzcd}$$

where α_1 has length at least 1. By Lemma 5.10, there is some W_2 such that:

$$\begin{tikzcd}
T_2 \arrow[r, "=", "\beta\top"'] \arrow[d, "{\Gamma,\, A\leq T_1}"', "\star"] & T_2 \arrow[d, "\star"', "{\Gamma,\, A\leq U_1}"] \\
V_2 \arrow[r, "\star", "\beta\top"'] & W_2.
\end{tikzcd}$$

So $W = \mathit{All}(A{\leq}W_1)\,W_2$ has the required property.

Case:
$$\frac{T_2 \longrightarrow_{\beta\top} U_2}{T = All(A{\leq}T_1)\,T_2 \longrightarrow_{\beta\top} All(A{\leq}T_1)\,U_2 = U}$$

By Lemma 5.6(1), V has the form $All(A{\leq}V_1)\,V_2$, with $T_1 \longrightarrow_{\Gamma} V_1$ and $T_2 \longrightarrow_{(\Gamma,\,A\leq T_1)} V_2$. We must show

$$\begin{tikzcd}
T_1 \arrow[rr, "\beta\top", "="'] \arrow[dd, "\Gamma"', "\star"] & & T_1 \arrow[dd, "\Gamma"', "\star"] & \\
& T_2 \arrow[rr, "\beta\top"] \arrow[dd, "{\Gamma,\,A\leq T_1}"', "\star"] & & U_2 \arrow[dd, dotted, "\star"', "{\Gamma,\,A\leq T_1}"] \\
V_1 \arrow[rr, "\beta\top", "="'] & & V_1 & \\
& V_2 \arrow[rr, dotted, "\beta\top", "\star"'] & & W_2,
\end{tikzcd}$$

which follows directly from the induction hypothesis.

Other cases:
Similarly, using parts (2) to (4) of 5.6. □

With this in hand, we can proceed to the main body of the strong normalization argument. Its two main steps are captured by this lemma and the next one.

5.12 Lemma [$\beta\top$ postponement]: If $T \longrightarrow_{\beta\top} U \longrightarrow_{\Gamma} X \longrightarrow^{\infty}_{\beta\top\Gamma} \ldots$, then there is some V_0 such that $T \longrightarrow_{\Gamma} V_0 \longrightarrow^{\infty}_{\beta\top\Gamma} \cdots$.

For the proof, we need a simple fact:

5.13 Fact: If $S \longrightarrow_{\beta\top} T \longrightarrow_{\Gamma} U$, then $S \longrightarrow_{\Gamma} U'$ for some U'. (That is, the redex that is contracted between T and U is a residual of a redex already present in S.

Proof: Since $\beta\top$-reduction cannot create a Γ-redex, the $\Gamma-$redex appearing in T must be a residual of a Γ-redex already appearing in S. □

Proof of Lemma 5.12: By Fact 5.13, there is some V_0 such that:

$$\begin{tikzcd}
T \arrow[r, "\Gamma"'] \arrow[d, "\beta\top"'] & V_0 \\
U \arrow[d, "\beta\top\Gamma"', "\infty"] & \\
\vdots &
\end{tikzcd}$$

By Lemma 5.11, there is some V_1 such that

$$\begin{tikzcd}
T \arrow[r, "\Gamma"'] \arrow[d, "\beta\top"'] & V_0 \arrow[d, "\beta\top\star"'] \\
U \arrow[r, "\Gamma\star"'] \arrow[d, "\beta\top\Gamma\star"', "\infty"] & V_1 \\
\vdots &
\end{tikzcd}$$

Since $U \longrightarrow_{\Gamma} X$, we can now apply Lemma 5.8:

$$\begin{tikzcd}
T \arrow[r, "\Gamma"'] \arrow[d, "\beta\top"'] & V_0 \arrow[d, "\beta\top\Gamma\star"'] \\
U \arrow[r, "\Gamma\star"'] \arrow[d, "\Gamma"'] & V_1 \arrow[d, "\Gamma\star"'] \\
X \arrow[r, "\Gamma\star"'] \arrow[d, "\beta\top\Gamma"', "\infty"] & V_2 \\
\vdots &
\end{tikzcd}$$

We can continue in this way, applying either 5.8 or 5.11 to successive elements of the infinite reduction beginning from X to obtain an infinite sequence of multi-step $\beta\top\Gamma$-reductions on the right:

$$\begin{tikzcd}
T \arrow[r, "\Gamma"'] \arrow[d, "\beta\top"'] & V_0 \arrow[d, "\beta\top\star"'] \\
U \arrow[r, "\Gamma\star"'] \arrow[d, "\Gamma"'] & V_1 \arrow[d, "\Gamma\star"'] \\
X \arrow[r, "\Gamma\star"'] \arrow[d, "\beta\top\Gamma"'] & V_2 \arrow[d, "\beta\top\Gamma\star"'] \\
U_3 \arrow[r, "\Gamma\star"'] \arrow[d, "\beta\top\Gamma"', "\infty"] & V_3 \arrow[d, "\beta\top\Gamma\star"', "\infty"] \\
\vdots & \vdots
\end{tikzcd}$$

But the sequence of reductions on the left must contain infinitely many $\beta\top$ steps (otherwise it would have an infinite Γ-tail), so Lemma 5.11 also tells us that infinitely many of the individual multi-step reductions on the right are nonempty. The reduction $T \longrightarrow_{\Gamma} V_0 \longrightarrow_{\beta\top\Gamma\star} V_1 \cdots \longrightarrow_{\beta\top\Gamma\star}$ is the desired one.

5.14 Proposition [Strong $\beta\top\Gamma$-normalization]: If S is well-kinded in Γ, then there is no infinite $\beta\top\Gamma$-reduction from S.

Proof: Assume, for a contradiction, that R is an infinite $\beta\top\Gamma$-reduction beginning from S. Let $R_0 = R$. Now repeat the following process as long as possible to construct a sequence R_1, R_2, $\ldots$ of infinite $\beta\top$-reductions, all starting from S:

> If R_i contains no Γ-reduction that is immediately preceded by a $\beta\top$-reduction, then stop. Otherwise, form R_{i+1} from R_i by using Lemma 5.12 repeatedly to move the first such Γ-reduction before any $\beta\top$-reduction.

Note that all of the R_i are infinite and that the first i steps in each R_i are all Γ-reductions. Now, there are two possibilities:

- The sequence of $R's$ eventually terminates, having reached some R_n in which all Γ-reductions precede the first $\beta\top$-reduction. But this means that R_n contains only Γ-reductions, contradicting Lemma 5.4, or has an infinite tail consisting only of $\beta\top$-reductions, contradicting Lemma 5.2.

- The sequence of $R's$ is infinite. But since each R_i begins with at least i Γ-reductions, we can use this to exhibit an infinite Γ-reduction beginning from S, contradicting Lemma 5.4. $\square$

6 Subtyping

Usually, in proof-theoretic analyses of calculi with subtyping, the subtyping relation itself presents the most challenging problems. This is also the case in $F^{\omega}_{\leq}$.

6.1 Proof Outline

Although the details of our development will be somewhat more involved, it is helpful to start by reviewing the standard argument [CG92, Ghe90, CMMS91, BCGS91, etc.] for the decidability of subtyping in the second-order system $F_{\leq}$:

1. Begin with an "original" presentation of the subtyping system that directly expresses its intended meaning, but which is not directly implementable.
2. Propose an alternative presentation of the same relation by a *syntax-directed* set of inference rules, in which the premises of each rule contain only metavariables whose values are uniquely determined by the form of the conclusion, and in which all the derivations of any given subtyping statement $\Gamma \vdash S \leq T$ must end with the same rule. (More precisely: more than one rule may be used to derive a given statement, as long as only one of them has premises whose applicability cannot be checked directly, without making any recursive calls.) This system can be implemented by a proof-search algorithm that will never have to guess or backtrack.
3. Check that this algorithm is indeed a decision procedure for the syntax-directed system by showing that proof search must terminate in finite time when started with any statement as its initial goal.
4. Show that the syntax-directed system is sound, in the sense that any subtyping statement derived by the algorithm is also derivable in the original system. This step is typically straightforward.
5. Finally, prove that the syntax-directed system is complete: that any statement derivable in the original system is also derivable by the algorithm. This step is where a deeper understanding is required.

The syntax-directed system may be viewed as a version of the original from which all "problematic" rules have been removed. In the case of $F_{\leq}$, there is just one such rule:

$$\frac{\Gamma \vdash S \leq U \qquad \Gamma \vdash U \leq T}{\Gamma \vdash S \leq T} \tag{S-Trans}$$

By analogy with proof theory, this rule is sometimes called the *cut rule* of the subtyping system: the type U appearing in the subderivations is cut out when moving to the conclusion. By analogy with the sequent calculus or the simply typed λ-calculus (c.f. [GLT89]), this cut rule can be almost completely eliminated by rewriting derivations.

But *not* completely. In one situation, transitivity is actually essential. Statements with variables on the left-hand side cannot, in general, be proved without using transitivity. For example,

$$C{\leq}Top(\star),\ B{\leq}C,\ A{\leq}B \vdash A \leq C$$

must be proved using two instances of S-TVar to establish the connections between A and B and between B and C, which are then joined by a single instance of transitivity. Thus, to eliminate S-Trans while retaining completeness, it is necessary to refine the treatment of variables, extending each instance of S-TVar with an "internal" use of transitivity:

$$\frac{\Gamma \vdash \Gamma(A) \leq T}{\Gamma \vdash A \leq T} \tag{S-TVar-Plus-Trans}$$

It is easy to see that replacing S-TVar with S-TVar-Plus-Trans in the original system does not affect its power. Moreover, S-Trans can now be completely eliminated without losing any derivable statements. The resulting subtyping algorithm (i.e., the recursive procedure obtained from the syntax-directed system by ordering overlapping rules so that the "easy" ones come first) is:

$$
\begin{array}{l}
\mathit{check}(\Gamma \vdash S \leq T) = \\
\quad \mathit{if}\ T \equiv \mathit{Top}(\star) \\
\quad\quad \mathit{then\ true} \\
\quad \mathit{else\ if}\ S \equiv T \\
\quad\quad \mathit{then\ true} \\
\quad \mathit{else\ if}\ S \equiv A \\
\quad\quad \mathit{then\ check}(\Gamma \vdash \Gamma(A) \leq T) \\
\quad \mathit{else\ if}\ S \equiv S_1 \to S_2\ \mathit{and}\ T \equiv T_1 \to T_2 \\
\quad\quad \mathit{then}\quad\quad \mathit{check}(\Gamma \vdash T_1 \leq S_1) \\
\quad\quad\quad\quad \mathit{and\ check}(\Gamma \vdash S_2 \leq T_2) \\
\quad \mathit{else\ if}\ S \equiv \mathit{All}(A{\leq}U)\, S_2\ \mathit{and}\ T \equiv \mathit{All}(A{\leq}U)\, T_2 \\
\quad\quad \mathit{then\ check}(\Gamma,\, A{\leq}U \vdash S_2 \leq T_2) \\
\quad \mathit{else} \\
\quad\quad \mathit{false}.
\end{array}
$$

The behavior of this algorithm reveals a great deal about the structure of the $F_{\leq}$ subtyping relation. The first cases deal with the easy rules for *Top* and reflexivity. The third case says that a statement of the form $\Gamma \vdash A \leq T$, where A is not identical to T and T is not *Top*, can *only* be true if A's upper bound is less that T. In other words, the region between A and its upper bound is empty: there are no types strictly greater than A and strictly less than $\Gamma(A)$.

Since this concept of "the smallest proper supertype of A" will also be crucial for our development, it is worth introducing some special notation for it. Write $A \uparrow_\Gamma \Gamma(A)$ for "A promotes to $\Gamma(S)$." We can then reformulate the enriched variable subtyping rule S-TVar-Plus-Trans as

$$
\frac{A \uparrow_\Gamma \Gamma(A) \qquad \Gamma \vdash S \leq T}{\Gamma \vdash A \leq T} \qquad \text{(S-Promote-TVar)}
$$

or, more generally, as

$$
\frac{S \uparrow_\Gamma U \qquad \Gamma \vdash U \leq T}{\Gamma \vdash S \leq T} \qquad \text{(S-Promote)}
$$

where the partial function $\uparrow_\Gamma$ is undefined except on variables. The subtyping algorithm becomes:

$$
\begin{array}{l}
\mathit{check}(\Gamma \vdash S \leq T) = \\
\quad \ldots \\
\quad \mathit{else\ if}\ S \uparrow_\Gamma U \\
\quad\quad \mathit{then\ check}(\Gamma \vdash U \leq T) \\
\quad \ldots
\end{array}
$$

Now, let us generalize these intuitions to the case of full $F^{\omega}_{\leq}$. Here, we encounter one new kind of situation in which transitivity plays an essential role. For example, in the context

$$
\Gamma = A{\leq}\mathit{Top}(\star),\ F{\leq}(\mathit{Fun}\,(B{:}\star)\,B),
$$

the statement $\Gamma \vdash F\ A \leq A$ is provable as follows (ignoring kinding):

$$
\dfrac{\dfrac{\dfrac{}{\Gamma \vdash F \leq (Fun(B{:}\star)B)}\ \text{S-TVar}}{\Gamma \vdash F\ A \leq (Fun(B{:}\star)B)\ A}\ \text{S-App} \qquad \dfrac{}{\Gamma \vdash (Fun(B{:}\star)B)\ A \leq A}\ \text{S-Conv}}{\Gamma \vdash F\ A \leq A}\ \text{S-Trans}
$$

The instance of transitivity in this derivation is again essential, but it is not an instance of the schema that motivated S-Tvar-Plus-Trans. In fact, it is possible to construct more involved examples where the instance of S-TVar is separated from the instance of S-Trans by arbitrarily many applications of S-App. This suggests the following generalization of the promotion relation:

6.1.1 Definition [Promotion]: The *promotion* of a type $A\ S_1 \ldots S_n$ in a well-formed context Γ is $\Gamma(A)\ S_1 \ldots S_n$, written $A\ S_1 \ldots S_n \uparrow_\Gamma \Gamma(A)\ S_1 \ldots S_n$.

With this relation and S-PROMOTE, both examples above can be derived without explicitly using S-TRANS.

To extend the algorithm *check* to full $F^{\omega}_{\leq}$, one thing obviously missing is a clause for type abstraction matching the pointwise subtyping rule S-ABS. We add one as follows:

$$
\begin{array}{l}
\mathit{check}(\Gamma \vdash S \leq T) = \\
\quad \ldots \\
\quad \mathit{else\ if}\ S \equiv \mathit{Fun}(A{:}K_1)\, S_2\ \mathit{and}\ T \equiv \mathit{Fun}(A{:}K_1)\, T_2 \\
\qquad \mathit{then\ check}(\Gamma,\, A{\leq}\,\mathit{Top}(K_1) \vdash S_2 \leq T_2)
\end{array}
$$

Surprisingly, we do not need a similar clause for application,

$$
\begin{array}{l}
\mathit{check}(\Gamma \vdash S \leq T) = \\
\quad \ldots \\
\quad \mathit{else\ if}\ S \equiv S_1\ U\ \mathit{and}\ T \equiv T_1\ U \\
\qquad \mathit{then\ check}(\Gamma \vdash S_1 \leq T_1)
\end{array}
$$

because its effect turns out to be covered by the promotion clause. But we do need to deal with the possibility of conversion; otherwise, for example, the statement $\Gamma \vdash (\mathit{Fun}\,(B{:}\star)\ B)S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2$ will not be derivable.

Clearly, we need to perform some reduction on the arguments before choosing which clause of the algorithm to apply. We can make life easy by simply *normalizing* the arguments before looking at them.[5]

Since the arguments to recursive calls in all of the clauses except promotion will remain in normal form if the original arguments are given in normal form, we only need to re-normalize in the promotion clause to preserve normality. The final algorithm, then, is:

$$
\begin{array}{l}
\mathit{check}(\Gamma \vdash S \leq T) = \\
\quad \mathit{check}^{!}(\Gamma \vdash S^{!} \leq T^{!}) \\
\\
\mathit{check}^{!}(\Gamma \vdash S \leq T) = \\
\quad \mathit{if}\ T \equiv \mathit{Top}(\mathit{Kind}_{\Gamma}(S)) \\
\qquad \mathit{then\ true} \\
\quad \mathit{else\ if}\ S \equiv T \\
\qquad \mathit{then\ true} \\
\quad \mathit{else\ if}\ S \uparrow_{\Gamma} U \\
\qquad \mathit{then\ check}^{!}(\Gamma \vdash U^{!} \leq T) \\
\quad \mathit{else\ if}\ S \equiv S_1 \rightarrow S_2\ \mathit{and}\ T \equiv T_1 \rightarrow T_2 \\
\qquad \mathit{then}\qquad \mathit{check}^{!}(\Gamma \vdash T_1 \leq S_1) \\
\qquad\qquad \mathit{and\ check}^{!}(\Gamma \vdash S_2 \leq T_2) \\
\quad \mathit{else\ if}\ S \equiv \mathit{All}(A{\leq}U)\, S_2\ \mathit{and}\ T \equiv \mathit{All}(A{\leq}U)\, T_2 \\
\qquad \mathit{then\ check}^{!}(\Gamma,\, A{\leq}U \vdash S_2 \leq T_2) \\
\quad \mathit{else\ if}\ S \equiv \mathit{Fun}(A{:}K_1)\, S_2\ \mathit{and}\ T \equiv \mathit{Fun}(A{:}K_1)\, T_2 \\
\qquad \mathit{then\ check}^{!}(\Gamma,\, A{\leq}\,\mathit{Top}(K_1) \vdash S_2 \leq T_2) \\
\quad \mathit{else} \\
\qquad \mathit{false.}
\end{array}
$$

Our task for the remainder of the section will be to show that this algorithm is sound and complete for the rules in Section 2.4. Our first step is a technical reformulation of the original system, which provides a convenient setting for the arguments to follow: we remove the general rule of conversion and regain its effect

[5] In a real implementation it is not desirable to fully normalize type expressions: this wastes time (in the vast majority of calls to the subtyping algorithm, the types being compared are identical) and results in the unnecessary expansion of type abbreviations, making the compiler's diagnostic output difficult for the programmer to understand. In practice, we reduce types only to weak head normal form, exposing only their outermost constructors at each step. The completeness of this modification rests on the observation that the reflexivity check in the algorithm can be restricted to type variables, applications, and the left-hand sides of quantifiers.

by generalizing each of the remaining rules to allow arbitrary reduction in the premises. For example, the rule for subtyping arrow types

$$\frac{\Gamma \vdash T_1 \leq S_1 \qquad \Gamma \vdash S_2 \leq T_2}{\Gamma \vdash S_1 \to S_2 \leq T_1 \to T_2}$$

becomes:

$$\frac{\begin{array}{c} S \longrightarrow^{\star}_{\beta\top} S_1 \to S_2 \qquad T \longrightarrow^{\star}_{\beta\top} T_1 \to T_2 \\ \Gamma \vdash T_1 \leq S_1 \qquad \Gamma \vdash S_2 \leq T_2 \end{array}}{\Gamma \vdash S \leq T}$$

It is not hard to show that this *reducing system* is equivalent to the original. Next, we introduce two important properties of certain derivations in the reducing system:

- A *cut-free* derivation is one with no instances of the rule of transitivity.
- A *strong* derivation is one in which every $\longrightarrow^{\star}_{\beta\top}$ reduction is actually a reduction to normal form.

Cut-free derivations are close to the form of those discovered by the algorithm *check*. Strength is a more technical property, which reduces the complexity of the case analyses required at some crucial points. Using these properties, the main facts that we need are:

1. The rule of transitivity can be eliminated from strong derivations.
2. The validity of subtyping is preserved by $\beta\top$-reduction in cut-free derivations.

From these, we can show the central theorem: any derivation in the reducing system can be transformed to a strong, cut-free derivation of the same statement. Finally, to obtain the algorithm, we observe that the rule of application can also be eliminated from strong, cut-free derivations.

6.2 The Reducing System

The main difference between the reducing system and the original subtyping system presented in Section 2 is that we remove the rule S-Conv and distribute its effects over the remaining rules in form of extra premises. We also replace the rule S-TVar by the more general rule of promotion, R-Promote.

6.2.1 Definition [Reducing System]:

$$\frac{S \longrightarrow^{\star}_{\beta\top} U \qquad T \longrightarrow^{\star}_{\beta\top} U}{\Gamma \vdash S \leq T} \tag{R-Refl}$$

$$\frac{\Gamma \vdash S \leq U \qquad \Gamma \vdash U \leq T \qquad \Gamma \vdash U \in K}{\Gamma \vdash S \leq T} \tag{R-Trans}$$

$$\frac{S \longrightarrow^{\star}_{\beta\top} U \qquad U \uparrow_{\Gamma} U' \qquad \Gamma \vdash U' \leq T}{\Gamma \vdash S \leq T} \tag{R-Promote}$$

$$\frac{T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K) \qquad \Gamma \vdash S \in K}{\Gamma \vdash S \leq T} \tag{R-Top}$$

$$\frac{\begin{array}{c} S \longrightarrow^{\star}_{\beta\top} S_1 \to S_2 \qquad T \longrightarrow^{\star}_{\beta\top} T_1 \to T_2 \\ \Gamma \vdash T_1 \leq S_1 \qquad \Gamma \vdash S_2 \leq T_2 \end{array}}{\Gamma \vdash S \leq T} \tag{R-Arrow}$$

$$\frac{S \longrightarrow^{\star}_{\beta\top} \mathit{All}(A{\leq}U)\, S_2 \qquad T \longrightarrow^{\star}_{\beta\top} \mathit{All}(A{\leq}U)\, T_2 \qquad \Gamma, A{\leq}U \vdash S_2 \leq T_2}{\Gamma \vdash S \leq T} \tag{R-All}$$

$$\frac{\begin{array}{c} S \longrightarrow^{\star}_{\beta\top} \mathit{Fun}\,(A{:}K)\, S' \qquad T \longrightarrow^{\star}_{\beta\top} \mathit{Fun}\,(A{:}K)\, T' \\ \Gamma, A{\leq}\mathit{Top}(K) \vdash S' \leq T' \end{array}}{\Gamma \vdash S \leq T} \tag{R-Abs}$$

$$\frac{S \longrightarrow^{\star}_{\beta\top} S'\ U \qquad T \longrightarrow^{\star}_{\beta\top} T'\ U \qquad \Gamma \vdash S' \leq T'}{\Gamma \vdash S \leq T} \tag{R-App}$$

6.2.2 Notation: To avoid confusion, we sometimes distinguish derivations in different systems by marking the turnstile symbol: $\vdash_{\mathcal{O}}$ for the original system, $\vdash_{\mathcal{R}}$ for the reducing system, $\vdash_{\mathcal{S}}$ for strong derivations in the reducing system, $\vdash_{\mathcal{C}}$ for cut-free derivations in the reducing system, and $\vdash_{\mathcal{CS}}$ for strong, cut-free derivations in the reducing system.

Our task for the remainder of Section 6.2 is to establish the equivalence of the reducing system and the original one. We begin by establishing some technical properties of the original.

6.2.3 Lemma [Kind invariance under promotion]: If $\Gamma \vdash S \in K$ and $S \Uparrow_{\Gamma} S'$ then $\Gamma \vdash S' \in K$.

6.2.4 Lemma [Promotion and Subtyping]: If $S \Uparrow_{\Gamma} S'$, then $\Gamma \vdash_{\mathcal{O}} S \leq S'$.

6.2.5 Lemma [Well-kinded subderivations]:

1. Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If d is a derivation of $\Gamma \vdash_{\mathcal{O}} S \leq T$ and d' is a derivation of $\Gamma' \vdash_{\mathcal{O}} S' \leq T'$, with d' a subderivation of d, then $\Gamma' \vdash S', T' \in K$ for some K.

2. Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If d is a derivation of $\Gamma \vdash_{\mathcal{R}} S \leq T$ and d' is a derivation of $\Gamma' \vdash_{\mathcal{R}} S' \leq T'$, with d' a subderivation of d, then $\Gamma' \vdash S', T' \in K$ for some K.

6.2.6 Lemma: The following rules are derivable:

$$\frac{\Gamma \vdash S,T \in K \qquad \Gamma \vdash_{\mathcal{O}} U \leq T \qquad \Gamma \vdash U \in K' \qquad U =_{\beta\top} S}{\Gamma \vdash_{\mathcal{O}} S \leq T} \qquad \text{(S-Conv(l))}$$

$$\frac{\begin{array}{c}\Gamma \vdash S,T \in K \qquad \Gamma \vdash S',T' \in K' \\ \Gamma \vdash_{\mathcal{O}} S' \leq T' \qquad S =_{\beta\top} S' \qquad T =_{\beta\top} T'\end{array}}{\Gamma \vdash_{\mathcal{O}} S \leq T} \qquad \text{(S-Conv(2))}$$

Proof: For S-Conv(1):

$$\dfrac{\dfrac{\Gamma \vdash_{\mathcal{O}} S \leq S \qquad \Gamma \vdash S \in K \qquad S =_{\beta\top} U}{\Gamma \vdash_{\mathcal{O}} S \leq U}\text{S-Conv} \qquad \Gamma \vdash_{\mathcal{O}} U \leq T \qquad \Gamma \vdash U \in K'}{\Gamma \vdash_{\mathcal{O}} S \leq T}\text{S-Trans}$$

For S-Conv(2), the proof is similar. □

6.2.7 Lemma [Equivalence]: The original and the intermediate subtyping systems are equivalent for well-kinded types: if $\Gamma \vdash S, T \in K$, then $\Gamma \vdash_{\mathcal{O}} S \leq T$ iff $\Gamma \vdash_{\mathcal{R}} S \leq T$.

Proof: By induction on derivations. □

For the remainder of Section 6, we will work exclusively within the reducing system.

6.3 Cut Elimination

We begin our analysis of the reducing system with a proof that R-Trans is inessential: any derivation ending with it can be rewritten as a derivation in the cut-free subsystem. To control the complexity of the combinatorial analysis, we do not show this property for arbitrary uses of transitivity, but only for uses of a restricted form: we consider only cut-terms in normal form and ask that the subderivations of the cut be strong. The following section will show that these conditions can always be achieved.

With these restrictions, the proof of cut elimination is a straightforward extension of standard proofs for the second-order fragment (c.f. [BCGS91, CG92, CMMS91, CP94]). We begin with one technical lemma.

6.3.1 Lemma: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash A\ T_1 \ldots T_n \in K_T$. If $\Gamma \vdash_{\mathcal{CS}} S \leq A\ T_1 \ldots T_n$, then this statement can be proved using a sequence of instances of R-PROMOTE preceded by a single instance of R-REFL.

$$\dfrac{S \longrightarrow^{!}_{\beta\top} \uparrow_\Gamma S_1 \qquad \dfrac{\dfrac{S_{n-1} \longrightarrow^{!}_{\beta\top} \uparrow_\Gamma S_n \qquad \dfrac{S_n \longrightarrow^{!}_{\beta\top} V \qquad A\ T_1 \ldots T_n \longrightarrow^{!}_{\beta\top} V}{\Gamma \vdash_{\mathcal{CS}} S_n \leq A\ T_1 \ldots T_n}}{\vdots}}{\Gamma \vdash_{\mathcal{CS}} S_1 \leq A\ T_1 \ldots T_n}}{\Gamma \vdash_{\mathcal{CS}} S \leq A\ T_1 \ldots T_n}$$

Proof: By induction on the given subtyping derivation. The R-REFL case is immediate; R-PROMOTE makes straightforward use of the induction hypothesis. R-APP uses the induction hypothesis to construct a derivation of the required shape for the left-hand sides of the application ($\Gamma \vdash_{\mathcal{CS}} S' \leq A\ T'_1 \ldots T'_{n-1}$); it is then easy to check that the right-hand side (T'_n) can be adjoined to all the steps in this derivation. □

6.3.2 Proposition [Cut elimination]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$ and $\Gamma \vdash U \in K_U$. If $\Gamma \vdash_{\mathcal{CS}} S \leq U$ and $\Gamma \vdash_{\mathcal{CS}} U \leq T$, where U is in normal form, then $\Gamma \vdash_{\mathcal{CS}} S \leq T$.

Proof: By induction on the combined size of the given subderivations. Proceed by a case analysis on the last rule in each.

Case: R-REFL on the left / anything on the right:
$$\dfrac{S \longrightarrow^{!}_{\beta\top} U \qquad U \longrightarrow^{!}_{\beta\top} U}{\Gamma \vdash_{\mathcal{CS}} S \leq U}$$

To prove $\Gamma \vdash_{\mathcal{CS}} S \leq T$, we can directly use the subderivations of $\Gamma \vdash_{\mathcal{CS}} U \leq T$, since $S \longrightarrow^{!}_{\beta\top} U$.

Case: Anything on the left / R-REFL on the right:
Similar.

Case: Anything on the left / R-TOP on the right:
$$\dfrac{T \longrightarrow^{!}_{\beta\top} \mathit{Top}(K) \qquad \Gamma \vdash_{\mathcal{CS}} U \in K}{\Gamma \vdash_{\mathcal{CS}} U \leq T}$$

By the well-kindedness of subderivations, we obtain $\Gamma \vdash_{\mathcal{CS}} S \leq T$ immediately by R-TOP.

Case: R-PROMOTE on the left / anything on the right:
$$\dfrac{S \longrightarrow^{!}_{\beta\top} \uparrow_\Gamma S' \qquad \Gamma \vdash_{\mathcal{CS}} S' \leq U}{\Gamma \vdash_{\mathcal{CS}} S \leq U}$$

By induction (using well-kindedness of subderivations) and R-PROMOTE.

Case: Anything on the left / R-PROMOTE on the right:
$$\dfrac{U = A\ U_1\ \ldots\ U_n \uparrow_\Gamma V \qquad \Gamma \vdash_{\mathcal{CS}} V \leq T}{\Gamma \vdash_{\mathcal{CS}} U \leq T}$$

By Lemma 6.3.1, we may assume that the derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq U$ consists of a sequence of instances of R-PROMOTE preceded by an instance of R-REFL:

$$\dfrac{S \longrightarrow^{!}_{\beta\top} S' \uparrow_\Gamma S_1 \qquad \dfrac{\dfrac{S_{n-1} \longrightarrow^{!}_{\beta\top} S'_{n-1} \uparrow_\Gamma S_n \qquad \dfrac{S_n \longrightarrow^{!}_{\beta\top} U \qquad A\ U_1 \ldots U_n = U}{\Gamma \vdash_{\mathcal{CS}} S_n \leq A\ U_1 \ldots U_n}}{\vdots}}{\Gamma \vdash_{\mathcal{CS}} S_1 \leq A\ U_1 \ldots U_n}}{\Gamma \vdash_{\mathcal{CS}} S \leq A\ U_1 \ldots U_n}$$

Replacing the final instance of R-Refl by an additional instance of R-Promote, we obtain

$$\dfrac{S \longrightarrow^{!}_{\beta\top} S' \uparrow_\Gamma S_1 \qquad \dfrac{\dfrac{S_{n-1} \longrightarrow^{!}_{\beta\top} S_{n-1} \uparrow_\Gamma S_n \qquad \dfrac{S_n \longrightarrow^{!}_{\beta\top} \uparrow_\Gamma V \qquad \Gamma \vdash_{\mathcal{CS}} V \leq T}{\Gamma \vdash_{\mathcal{CS}} S_n \leq T}}{\vdots}}{\Gamma \vdash_{\mathcal{CS}} S_1 \leq T}}{\Gamma \vdash_{\mathcal{CS}} S \leq T}$$

as desired.

This takes care of 29 of the 49 cases. The remaining cases are listed in the following table.

$\Gamma \vdash_{\mathcal{CS}} U \leq T$

$\Gamma \vdash_{\mathcal{CS}} S \leq U$

	R-Arrow	R-All	R-Abs	R-App
R-Top	×	×	×	×
R-Arrow	√	×	×	×
R-All	×	√	×	×
R-Abs	×	×	√	×
R-App	×	×	×	√

The cases marked √ are dealt with individually below. Those marked × can never occur, since the two rules in question would place incompatible constraints on the form of U. (This can easily be checked by inspecting the rules; the assumption that the given derivations are strong is needed in several places.)

Case R-All:

$$\dfrac{S \longrightarrow^{!}_{\beta\top} All(A{\leq}U_1)\, S_2 \quad U = All(A{\leq}U_1)\, U_2 \quad \Gamma,\, A{\leq}U_1 \vdash S_2 \leq U_2}{\Gamma \vdash_{\mathcal{CS}} S \leq U} \qquad \dfrac{U = All(A{\leq}U_1)\, U_2 \quad T \longrightarrow^{!}_{\beta\top} All(A{\leq}U_1)\, T_2 \quad \Gamma,\, A{\leq}U_1 \vdash U_2 \leq T_2}{\Gamma \vdash_{\mathcal{CS}} U \leq T}$$

By the well-kindedness of subderivations, the induction hypothesis applies, giving:

$$\dfrac{S \longrightarrow^{!}_{\beta\top} All(A{\leq}U_1)\, S_2 \qquad T \longrightarrow^{!}_{\beta\top} All(A{\leq}U_1)\, T_2 \qquad \Gamma,\, A{\leq}U_1 \vdash S_2 \leq T_2}{\Gamma \vdash_{\mathcal{CS}} S \leq T}$$

Case R-Arrow, R-Abs**:**
Similar.

Case R-App:

$$\dfrac{S \longrightarrow^{!}_{\beta\top} S'\ V \qquad U = U'\ V \qquad \Gamma \vdash_{\mathcal{CS}} S' \leq U'}{\Gamma \vdash_{\mathcal{CS}} S \leq U} \qquad \dfrac{U = U'\ V \qquad T \longrightarrow^{!}_{\beta\top} T'\ V \qquad \Gamma \vdash_{\mathcal{CS}} U' \leq T'}{\Gamma \vdash_{\mathcal{CS}} \Gamma \vdash U \leq T}$$

By the well-kindedness of subderivations, the induction hypothesis applies, giving:

$$\dfrac{S \longrightarrow^{!}_{\beta\top} S'\ V \qquad T \longrightarrow^{!}_{\beta\top} T'\ V \qquad \Gamma \vdash_{\mathcal{CS}} S' \leq T'}{\Gamma \vdash_{\mathcal{CS}} S \leq T}$$

□

6.4 Reduction and Subtyping

The main task of this subsection is to show that, for cut-free derivations, $\beta\top$-reduction in types does not interfere with the subtyping judgement.

The cornerstone of the argument is a substitution lemma saying (informally) that if $S \leq T$ then $[V/A]S \leq [V/A]T$. From this, we can show that the reduction of an outermost redex on either the left-hand or the right-hand side of a subtyping statement preserves its derivability. As in the proof of Church-Rosser in Section 3, we extend these properties to a proof of the preservation of subtyping under arbitrary multi-step reduction by passing through an intermediate step where we show it for one-step parallel reduction.

In outline, then, the major steps are as follows:

$$\begin{tikzcd}
 & \text{reduction (6.4.9)} & \\
 & \text{parallel reduction (6.4.8)} \arrow[u] & \\
\text{outer reduction (left) (6.4.7)} \arrow[ur] & & \text{outer reduction (right) (6.4.5)} \arrow[ul] \\
 & \text{substitution (6.4.3)} \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

We begin with two technical lemmas:

6.4.1 Lemma [Expansion preserves subtyping]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} S' \leq T'$ with $S \longrightarrow^{\star}_{\beta\top} S'$ and $T \longrightarrow^{\star}_{\beta\top} T'$, then $\Gamma \vdash_{\mathcal{C}} S \leq T$.

Proof: By inspection of the rules, using the properties of kinding and reduction. $\square$

6.4.2 Lemma [Maximality of Top]: Suppose $\Gamma \vdash \mathit{Top}(K)\ S_1 \ldots S_n \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K)\ S_1 \ldots S_n \leq T$, then $T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K_T)$.

Proof: Straightforward induction, using the properties of kinding and reduction. $\square$

Now we come to the key result of this subsection: the preservation of subtyping under substitution.

6.4.3 Lemma [Substitution preserves subtyping]: Let $\Gamma = \Gamma_1, A{\leq}\mathit{Top}(K), \Gamma_2$ and $\Gamma' = \Gamma_1, [V/A]\Gamma_2$. Suppose that $\Gamma_1 \vdash V \in K$ and that $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma' \vdash_{\mathcal{C}} [V/A]S \leq [V/A]T$.

Proof: The cases other than R-Promote are straightforward; we give the argument for R-All as an example.[6]

Case R-All:

$$\frac{S \longrightarrow^{\star}_{\beta\top} \mathit{All}(A'{\leq}U)\ S_2 \qquad T \longrightarrow^{\star}_{\beta\top} \mathit{All}(A'{\leq}U)\ T_2 \qquad \Gamma, A'{\leq}U \vdash_{\mathcal{C}} S_2 \leq T_2}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma 3.3(3), we have

$$[V/A]S \longrightarrow^{\star}_{\beta\top} \mathit{All}(A'{\leq}[V/A]U)\ [V/A]S_2$$
$$[V/A]T \longrightarrow^{\star}_{\beta\top} \mathit{All}(A'{\leq}[V/A]U)\ [V/A]T_2.$$

By the well-kinded subderivations lemma and the induction hypothesis, $\Gamma', A'{\leq}[V/A]U \vdash_{\mathcal{C}} [V/A]S_2 \leq [V/A]T_2$. Now R-All applies as follows:

$$\frac{[V/A]S \longrightarrow^{\star}_{\beta\top} \mathit{All}(A'{\leq}[V/A]U)\ [V/A]S_2 \qquad [V/A]T \longrightarrow^{\star}_{\beta\top} \mathit{All}(A'{\leq}[V/A]U)\ [V/A]T_2 \qquad \Gamma', A'{\leq}[V/A]U \vdash_{\mathcal{C}} [V/A]S_2 \leq [V/A]T_2}{\Gamma' \vdash_{\mathcal{C}} [V/A]S \leq [V/A]T}$$

[6] Note that the more general form of this property, in which $\Gamma(A)$ is allowed to be any supertype of V, would be much more difficult to prove. Here we obtain a straightforward proof by considering only the form that will actually be required later: in the critical case — the one for R-Promote — the fact that $\Gamma(A) = \mathit{Top}(K)$ allows a direct argument using Lemma 6.4.2.

Case R-PROMOTE:

$$\frac{S \longrightarrow^{\star}_{\beta\top} S' \qquad S' \uparrow_{\Gamma} U \qquad \Gamma \vdash_{\mathcal{C}} U \leq T}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

The definition of promotion gives $S' = A'\ U_1 \dots U_n$ for some $n \geq 0$, and $U = \Gamma(A')\ U_1 \dots U_n$. Now there are two subcases to consider:

Subcase: $A' \neq A$

By induction, $\Gamma' \vdash_{\mathcal{C}} [V/A](\Gamma(A')\ U_1 \dots U_n) \leq [V/A]T$. Now,

$$[V/A](\Gamma(A')\ U_1 \dots U_n) = ([V/A](\Gamma(A')))\ [V/A]U_1 \dots [V/A]U_n = \Gamma'(A')\ [V/A]U_1 \dots [V/A]U_n,$$

and by Lemma 3.3(3), $[V/A]S \longrightarrow^{\star}_{\beta\top} A'\ [V/A]U_1 \dots [V/A]U_n$. So, by the definition of promotion, we can apply R-PROMOTE to obtain the desired result as follows:

$$\frac{\begin{array}{c}[V/A]S \longrightarrow^{\star}_{\beta\top} A'\ [V/A]U_1 \dots [V/A]U_n \uparrow_{\Gamma'} [V/A](\Gamma(A')\ U_1 \dots U_n) \\ \Gamma' \vdash_{\mathcal{C}} [V/A](\Gamma(A')\ U_1 \dots U_n) \leq [V/A]T\end{array}}{\Gamma' \vdash_{\mathcal{C}} [V/A]S \leq [V/A]T}$$

Subcase: $A' = A$

Since $\Gamma(A) = \mathit{Top}(K)$, we have $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K)\ U_1 \dots U_n \leq T$. By the well-kinded subderivations lemma (6.2.5), U is well kinded in Γ. So by Lemma 6.4.2, we have $T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K_T)$. By Lemma 3.3, also $[V/A]T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K_T)$. By the well-kindedness of subderivations, $\Gamma \vdash S \in K_T$, so by the fact that substitution preserves kinding (Lemma 4.11), rule R-TOP

$$\frac{[T/A]T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K_T) \qquad \Gamma' \vdash [V/A]S \in K_T}{\Gamma' \vdash_{\mathcal{C}} [V/A]S \leq [V/A]T}$$

gives us the desired result. □

The next lemma introduces a technical property needed for the following one: a subtype of a Fun-type is either a Fun-type itself or can be promoted to one in a finite number of steps.

6.4.4 Lemma [Fun-right]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T$, then this statement can be derived by a sequence of instances of R-PROMOTE preceded by one instance of R-FUN:

$$\dfrac{S \longrightarrow^{\star}_{\beta\top} S' \uparrow_{\Gamma} S_1 \qquad \dfrac{\vdots}{\Gamma \vdash_{\mathcal{C}} S_1 \leq \mathit{Fun}(A{:}K)T}}{\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T}$$

$$\vdots$$

$$\dfrac{S_{n-1} \longrightarrow^{\star}_{\beta\top} S'_{n-1} \uparrow_{\Gamma} S_n \qquad \dfrac{\begin{array}{c}S_n \longrightarrow^{\star}_{\beta\top} \mathit{Fun}(A{:}K)S'_n \\ \mathit{Fun}(A{:}K)T \longrightarrow^{\star}_{\beta\top} \mathit{Fun}(A{:}K)T' \\ \Gamma, A{\leq}\mathit{Top}(K) \vdash_{\mathcal{C}} S'_n \leq T'\end{array}}{\Gamma \vdash_{\mathcal{C}} S_n \leq \mathit{Fun}(A{:}K)T}}{\vphantom{x}}$$

Proof: By induction on a derivation of $\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T$. By the form of the right-hand side and the fact that the derivation is cut free, there are three cases to consider. (Note, in passing, that this is one point where we crucially depend on the absence of cut: this argument fails on arbitrary reducing derivations.)

Case R-REFL:

$$\frac{S \longrightarrow^{\star}_{\beta\top} U \qquad \mathit{Fun}(A{:}K)T \longrightarrow^{\star}_{\beta\top} U}{\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T}$$

By definition of $\longrightarrow^{\star}_{\beta\top}$, the type U must be of the form $\mathit{Fun}(A{:}K)T'$, with $T \longrightarrow^{\star}_{\beta} T'$. So, by R-ABS and R-REFL:

$$\frac{\begin{array}{c}S \longrightarrow^{\star}_{\beta\top} \mathit{Fun}(A{:}K)T' \qquad \mathit{Fun}(A{:}K)T \longrightarrow^{\star}_{\beta\top} \mathit{Fun}(A{:}K)T' \\ \Gamma, A \leq \mathit{Top}(K) \vdash_{\mathcal{C}} T' \leq T'\end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T}$$

Case R-ABS:
Immediate.

Case R-PROMOTE:

$$\frac{S \longrightarrow^{\star}_{\beta\top} S' \uparrow_\Gamma U \qquad \Gamma \vdash_{\mathcal{C}} U \leq Fun(A{:}K)T}{\Gamma \vdash_{\mathcal{C}} S \leq Fun(A{:}K)T}$$

By the well-kinded subderivations lemma, the induction hypothesis, and R-PROMOTE. □

6.4.5 Lemma [Outer reduction on the right]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash (Fun(A{:}K)T)\ U \in K_T$. If $\Gamma \vdash_{\mathcal{C}} S \leq (Fun(A{:}K)T)\ U$, then $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T$.

Proof: By induction on a proof of $\Gamma \vdash_{\mathcal{C}} S \leq (Fun(A{:}K)T)\ U$. The only difficult case is R-APP, which we give in full below. R-REFL uses the Church-Rosser property. R-PROMOTE uses only the induction hypothesis. All the other cases follow a common pattern: we show just R-ALL.

Case R-ALL:

$$\frac{S \longrightarrow^{\star}_{\beta\top} All(A'{\leq}V)\, S_2 \qquad (Fun(A{:}K)T)\ U \longrightarrow^{\star}_{\beta\top} All(A'{\leq}V)\, T_2 \qquad \Gamma, A'{\leq}V \vdash_{\mathcal{C}} S_2 \leq T_2}{\Gamma \vdash_{\mathcal{C}} S \leq (Fun(A{:}K)T)\ U}$$

By the outermost reduction corollary (3.5), $[U/A]T \longrightarrow^{\star}_{\beta\top} All(A'{\leq}V)\, T_2$. The result follows immediately by R-ALL.

Case R-APP:

There are two cases to consider, depending on whether the redex $(Fun(A{:}K)T)\ U$ is itself contracted at some point in the reduction sequence $(Fun(A{:}K)T)\ U \longrightarrow^{\star}_{\beta\top} \cdots$.

Subcase:

$$\frac{(Fun(A{:}K)\ T)\ U \longrightarrow^{\star}_{\beta\top} (Fun(A{:}K)T')\ U' \longrightarrow_{\beta\top} [U'/A]T' \longrightarrow^{\star}_{\beta\top} T_1\ V \qquad S \longrightarrow^{\star}_{\beta\top} W\ V \qquad \Gamma \vdash_{\mathcal{C}} W \leq T_1}{\Gamma \vdash_{\mathcal{C}} S \leq (Fun(A{:}K)T)\ U}$$

Since $[U/A]T \longrightarrow^{\star}_{\beta\top} [U'/A]T' \longrightarrow^{\star}_{\beta\top} T_1\ V$ (by Lemma 3.3(3)), the result follows immediately from R-APP.

Subcase:

$$\frac{S \longrightarrow^{\star}_{\beta\top} W\ U' \qquad (Fun(A{:}K)\ T)\ U \longrightarrow^{\star}_{\beta\top} (Fun(A{:}K)T')\ U' \qquad \Gamma \vdash_{\mathcal{C}} W \leq Fun(A{:}K)T'}{\Gamma \vdash_{\mathcal{C}} S \leq (Fun(A{:}K)T)\ U}$$

Here we cannot directly use the subderivation of $\Gamma \vdash_{\mathcal{C}} W \leq Fun(A{:}K)T'$. Intuitively, we must "look inside" this derivation to find an inner subderivation in which T' appears by itself on the right-hand side, and use this to rebuild a subderivation ending with $[U/A]T$ on the right. (More precisely, the inner subderivation will have T'' on the right, where $T' \longrightarrow^{\star}_{\beta\top} T''$.) This we accomplish as follows.

First, we use the Fun-right lemma (6.4.4) to obtain a derivation of $\Gamma \vdash_{\mathcal{C}} W \leq Fun(A{:}K)T'$ in a very rigid form: a sequence of n instances of R-PROMOTE ending with an instance of R-ABS:

$$\cfrac{W \longrightarrow^{\star}_{\beta\top} \uparrow_\Gamma W_1 \qquad \cfrac{\cfrac{W_{n-1} \longrightarrow^{\star}_{\beta\top} \uparrow_\Gamma W_n \qquad \cfrac{W_n \longrightarrow^{\star}_{\beta\top} Fun(A{:}K)X \qquad Fun(A{:}K)T' \longrightarrow^{\star}_{\beta\top} Fun(A{:}K)T'' \qquad \Gamma, A{\leq}Top(K) \vdash_{\mathcal{C}} X \leq T''}{\Gamma \vdash_{\mathcal{C}} W_n \leq Fun(A{:}K)T'}}{\vdots}}{\Gamma \vdash_{\mathcal{C}} W_1 \leq Fun(A{:}K)T'}}{\Gamma \vdash_{\mathcal{C}} W \leq Fun(A{:}K)T'}$$

From the instance of R-ABS at the top, we obtain $\Gamma \vdash_{\mathcal{C}} [U'/A]X \leq [U'/A]T''$ by the substitution lemma (6.4.3). (To check that the lemma applies, we need the following observations: (1) $\Gamma \vdash U' \in K$ by the assumption $\Gamma \vdash (Fun(A{:}K)\,T)U \in K_T$, the fact that $U \longrightarrow^{\star}_{\beta\top} U'$, the generation lemma for types, and subject reduction. (2) X and T'' are both well-kinded in Γ, $A{\leq}Top(K)$, by the well-kindedness of subderivations.)

At this point, we have accomplished a substitution operation, but not quite the one we need. We must now work backwards to the desired result.

From $\Gamma \vdash_{\mathcal{C}} [U'/A]X \leq [U'/A]T''$ and the fact that $[U/A]T \longrightarrow^{\star}_{\beta\top} [U'/A]T''$ (Lemma 3.3), we use the expansion lemma (6.4.1) to obtain $\Gamma \vdash_{\mathcal{C}} [U'/A]X \leq [U/A]T$, after remarking that $[U'/A]X$ and $[U/A]T$ are well-kinded by subject reduction. This gives us what we need on the right-hand side; now we turn to fixing the left-hand side.

$W_n\ U'$ is well-kinded and $W_n\ U' \longrightarrow^{\star}_{\beta\top} (\mathit{Fun}(A{:}K)\ X)\ U' \longrightarrow_{\beta\top} [U'/A]X$, so by the expansion lemma, $\Gamma \vdash_{\mathcal{C}} W_n\ U' \leq [U/A]T$. We use this as the starting point for a new sequence of R-Promote steps where an application to U' has been added to each intermediate term:

$$\dfrac{S \longrightarrow^{\star}_{\beta\top} W\ U' \longrightarrow^{\star}_{\beta\top}\ \Uparrow_\Gamma W_1\ U' \qquad \dfrac{\dfrac{W_{n-1}\ U' \longrightarrow^{\star}_{\beta\top}\ \Uparrow_\Gamma W_n\ U' \qquad \Gamma \vdash_{\mathcal{C}} W_n\ U' \leq [U/A]T}{\vdots}}{\Gamma \vdash_{\mathcal{C}} W_1\ U' \leq [U/A]T}}{\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T}$$

The conclusion of this derivation is the desired statement. □

We proceed for the left-hand side of $\leq$ in the same way as we have done for the righthand side.

6.4.6 Lemma [Fun-left]: Suppose $\Gamma \vdash \mathit{Fun}(A{:}K)S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S \leq T$, then one of the following cases holds:

1. $T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K_T)$; or
2. $\mathit{Fun}(A{:}K)S \longrightarrow^{\star}_{\beta\top} \mathit{Fun}(A{:}K)S'$ and $T \longrightarrow^{\star}_{\beta\top} \mathit{Fun}(A{:}K)T'$, with $\Gamma, A{\leq}\mathit{Top}(K) \vdash_{\mathcal{C}} S' \leq T'$.

Proof: By inspection of the rules. By the form of the left-hand side of the statement and the fact that we consider only cut-free derivations, there are only three cases to consider:

Case R-Refl:

$$\dfrac{\mathit{Fun}(A{:}K)S \longrightarrow^{\star}_{\beta\top} U \qquad T \longrightarrow^{\star}_{\beta\top} U}{\Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S \leq T}$$

U must be of the form $\mathit{Fun}(A{:}K)S'$ where $S \longrightarrow^{\star}_{\beta\top} S'$ so we can conclude:

$$\dfrac{S' \longrightarrow^{\star}_{\beta\top} S' \qquad S' \longrightarrow^{\star}_{\beta\top} S'}{\Gamma, A{\leq}\mathit{Top}(K) \vdash_{\mathcal{C}} S' \leq S'}$$

Case R-Top, R-Abs**:**
By case 1 and 2, respectively. □

6.4.7 Lemma [Outer reduction on the left]: Suppose $\Gamma \vdash (\mathit{Fun}(A{:}K)S)\ U \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K)S)\ U \leq T$, then $\Gamma \vdash_{\mathcal{C}} [U/A]S \leq T$.

Proof: By induction. Again, R-App is the only difficult case. If the reduction starting from $(\mathit{Fun}(A{:}K)S)\ U$ reduces the outer redex at some point, then this case proceeds as in the analogous case on the right-hand side (6.4.5). Otherwise, we have:

$$\dfrac{(\mathit{Fun}(A{:}K)\ S)\ U \longrightarrow^{\star}_{\beta\top} (\mathit{Fun}(A{:}K)S')\ U' \qquad T \longrightarrow^{\star}_{\beta\top} W\ U' \atop \Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S' \leq W}{\Gamma \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K)S)\ U \leq T}$$

By the well-kindedness of subderivations, both $\mathit{Fun}(A{:}K)\ S'$ and W are well kinded. By the Fun-left lemma, there are two cases to consider:

1. We are given $W \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K{\to}K_T)$. Since $\Gamma \vdash [U/A]S \in K_T$ (by subject reduction and uniqueness of kinding) and $T \longrightarrow^{\star}_{\beta\top} W\ V \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K{\to}K_T)\ V \longrightarrow_{\beta\top} \mathit{Top}(K_T)$, we can conclude $\Gamma \vdash_{\mathcal{C}} [U/A]S \leq T$ by R-Top.

2. We are given

$$\begin{array}{l} \mathit{Fun}\,(A{:}K)\,S' \longrightarrow^{\star}_{\beta} \mathit{Fun}\,(A{:}K)\,S'' \\ W \longrightarrow^{\star}_{\beta\top} \mathit{Fun}\,(A{:}K)\,Y \\ \Gamma,\, A{\leq}\,\mathit{Top}(K) \vdash_{\mathcal{C}} S'' \leq Y. \end{array}$$

   Both S'' and Y are well kinded in $\Gamma,\, A{\leq}\,\mathit{Top}(K)$ (by well-kindedness of subderivations, subject reduction, and the generation lemma), so $\Gamma \vdash_{\mathcal{C}} [U'/A]S' \leq [U'/A]Y$ by the substitution lemma (6.4.3).

   As in the analogous case on the right, we have now accomplished a substitution operation, but not quite the one we need. We must again work backwards to the desired result.

   From $\Gamma \vdash_{\mathcal{C}} [U'/A]S' \leq [U'/A]Y$ and the fact that $[U/A]S \longrightarrow^{\star}_{\beta\top} [U'/A]S''$ (Lemma 3.3), we can use the expansion lemma to obtain $\Gamma \vdash_{\mathcal{C}} [U/A]S \leq [U'/A]Y$. (Note that $[U/A]S$ is well kinded by subject reduction and $[U'/A]Y$ by subject reduction and the generation lemma.) Then we have $T \longrightarrow^{\star}_{\beta\top} W\ V \longrightarrow^{\star}_{\beta\top} (\mathit{Fun}\,(A{:}K)\,Y)\ U' \longrightarrow_{\beta\top} [U'/A]Y$, which yields the desired result by the expansion lemma. □

Using the two outer reduction lemmas, we can do one reduction step at the outside of a type while preserving the subtyping judgement. The next step is to generalize this to an arbitrary reduction. For this purpose, we choose the relation $\twoheadrightarrow_{\beta\top}$.

6.4.8 Proposition [Parallel reduction preserves subtyping]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$.

1. If $\Gamma \vdash_{\mathcal{C}} S \leq T$ with $S \twoheadrightarrow_{\beta\top} S'$ and $\Gamma \twoheadrightarrow_{\beta\top} \Gamma'$, then $\Gamma' \vdash_{\mathcal{C}} S' \leq T$.
2. If $\Gamma \vdash_{\mathcal{C}} S \leq T$ with $T \twoheadrightarrow_{\beta\top} T'$ and $\Gamma \twoheadrightarrow_{\beta\top} \Gamma'$, then $\Gamma' \vdash_{\mathcal{C}} S \leq T'$.

(Parallel reduction of contexts, written $\Gamma \twoheadrightarrow_{\beta\top} \Gamma'$, is the pointwise extension of parallel reduction of types.)

Proof: By simultaneous induction on derivations. The only difficult cases are R-App and R-Promote. We show only the arguments for part (1), where $S \twoheadrightarrow_{\beta\top} S'$; part (2) is similar except in the case R-Promote, which is easier.

Case R-Refl:

$$\frac{S \longrightarrow^{\star}_{\beta\top} U \qquad T \longrightarrow^{\star}_{\beta} U}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma 3.8 there is a type U' with $U \twoheadrightarrow_{\beta\top} U'$ and $S' \longrightarrow^{\star}_{\beta\top} U'$ as well as $T \longrightarrow^{\star}_{\beta\top} U'$. The result follows by R-Refl.

Case R-Top:

$$\frac{T \longrightarrow^{\star}_{\beta\top} \mathit{Top}(K') \qquad \Gamma \vdash S \in K'}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By subject reduction and uniqueness of kinding we have $K' = K_T$. So by subject reduction, the well-kindedness of subderivations, and Lemma 4.12, the result follows by R-Top.

Case R-Arrow:

$$\frac{\begin{array}{cc} S \longrightarrow^{\star}_{\beta\top} S_1 {\to} S_2 & T \longrightarrow^{\star}_{\beta\top} T_1 {\to} T_2 \\ \Gamma \vdash_{\mathcal{C}} T_1 \leq S_1 & \Gamma \vdash_{\mathcal{C}} S_2 \leq T_2 \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma 3.8 and the definition of parallel reduction (3.1), there are types S_1' and S_2' with $S_1 \twoheadrightarrow_{\beta\top} S_1'$ and $S_2 \twoheadrightarrow_{\beta\top} S_2'$ and:

$$\begin{tikzcd} S \arrow[r, two heads, "\beta\top"'] \arrow[d, "\star\,\beta\top"] & S' \arrow[d, "\star\,\beta\top"] \\ S_1 \to S_2 \arrow[r, two heads, "\beta\top"'] & S_1' \to S_2' \end{tikzcd}$$

Since the S_i and T_i are well-kinded, the result follows by induction.

Case R-ALL:

$$\frac{S \longrightarrow^{\star}_{\beta\top} \mathit{All}(A{\leq}U)V \qquad T \longrightarrow^{\star}_{\beta\top} \mathit{All}(A{\leq}U)W \qquad \Gamma, A{\leq}U \vdash_C V \leq W}{\Gamma \vdash_C S \leq T}$$

By Lemma 3.8 and the definition of parallel reduction (3.1), there are types U' and V' with $U \twoheadrightarrow_{\beta\top} U'$ and $V \twoheadrightarrow_{\beta\top} V'$ and:

$$\begin{tikzcd}
S \arrow[r, two heads, "\beta\top"'] \arrow[d, "\star"', "\beta\top"] & S' \arrow[d, "\star"', "\beta\top"] \\
\mathit{All}(A{\leq}U)V \arrow[r, two heads, "\beta\top"'] & \mathit{All}(A{\leq}U')V'
\end{tikzcd}$$

Since the V and W are well-kinded in $\Gamma, A{\leq}U$, the result follows by induction:

$$\frac{S' \longrightarrow^{\star}_{\beta\top} \mathit{All}(A{\leq}U')V' \qquad T \longrightarrow^{\star}_{\beta\top} \mathit{All}(A{\leq}U')W \qquad \Gamma', A{\leq}U' \vdash_C V' \leq W}{\Gamma' \vdash_C S' \leq T}$$

Case R-ALL, R-ABS:
Similar.
Case R-PROMOTE:

$$\frac{S \longrightarrow^{\star}_{\beta\top} W \qquad W \uparrow_{\Gamma} U \qquad \Gamma \vdash_C U \leq T}{\Gamma \vdash_C S \leq T}$$

By the definition of the promotion relation, we have $W = A\ S_1 \ldots S_n \uparrow_{\Gamma} \Gamma(A)\ S_1 \ldots S_n = U$. By Lemma 3.8 and the definition of $\twoheadrightarrow_{\beta\top}$, there are $S'_1 \ldots S'_n$ with $S_i \twoheadrightarrow_{\beta\top} S'_i$ and:

$$\begin{tikzcd}
S \arrow[r, two heads, "\beta\top"'] \arrow[d, "\star"', "\beta\top"] & S' \arrow[d, "\star"', "\beta\top"] \\
A\ S_1 \ldots S_n \arrow[r, two heads, "\beta\top"'] & A\ S'_1 \ldots S'_n
\end{tikzcd}$$

Let $U' = \Gamma'(A)\ S'_1 \ldots S'_n$. By well-kindedness of subderivations and subject reduction, U and U' are well kinded. The result follows by induction, the fact that $\Gamma(A)\ S_1 \ldots S_n \twoheadrightarrow_{\beta\top} \Gamma'(A)\ S'_1 \ldots S'_n$, and R-PROMOTE.
Case R-APP:

$$\frac{S \longrightarrow^{\star}_{\beta\top} U\ W \qquad T \longrightarrow^{\star}_{\beta\top} V\ W \qquad \Gamma \vdash_C U \leq V}{\Gamma \vdash_C S \leq T}$$

By Lemma 3.8, there is some X such that:

$$\begin{tikzcd}
S \arrow[r, two heads, "\beta\top"'] \arrow[d, "\star"', "\beta\top"] & S' \arrow[d, "\star"', "\beta\top"] \\
U\ W \arrow[r, two heads, "\beta\top"'] & X
\end{tikzcd}$$

By subject reduction, $\Gamma \vdash U\ W \in K_S$ and $\Gamma \vdash V\ W \in K_T$. By the generation lemma, this implies that U and V are well kinded. Continue by cases on the form of U. (The interesting one is when U is a type operator.)

Subcase: $U = A$ or $U = U_1\ U_2$
By the definition of parallel reduction, X must have the form $U'\ W'$ with $U \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} U'$ and $W \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} W'$. By subject reduction, U' is well kinded. The result then follows by induction and R-App:

$$\frac{S' \longrightarrow^{\star}_{\beta\top} U'\ W' \qquad T \longrightarrow^{\star}_{\beta\top} V\ W \longrightarrow^{\star}_{\beta\top} V\ W' \qquad \Gamma' \vdash_{\mathcal{C}} U' \leq V}{\Gamma' \vdash_{\mathcal{C}} S' \leq T}$$

Subcase: $U = Top(K')$

$$\frac{\begin{array}{c} S \longrightarrow^{\star}_{\beta\top} Top(K')\ W \qquad T \longrightarrow^{\star}_{\beta\top} V\ W \\ \Gamma \vdash_{\mathcal{C}} Top(K') \leq V \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma 6.4.2, $V \longrightarrow^{\star}_{\beta\top} Top(K')$. Since T has kind K_T, so does $Top(K')\ W$, and, by Lemma 4.10, $T \longrightarrow^{\star}_{\beta\top} Top(K')\ W \longrightarrow_{\beta\top} Top(K_T)$. The result then follows by R-Top (using subject reduction, well-kindedness of subderivations, and lemma 4.12).

$$\frac{T \longrightarrow^{\star}_{\beta\top} Top(K_T) \qquad \Gamma' \vdash S' \in K_T}{\Gamma' \vdash_{\mathcal{C}} S' \leq T}$$

Subcase: $U = Fun(A{:}K)U_1$
We have $(Fun(A{:}K)U_1)\ W \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} X$. By the definition of parallel reduction, X can have one of two forms:

Subsubcase: $X = (Fun(A{:}K)U_1')\ W'$
$Fun(A{:}K)U_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} Fun(A{:}K)U_1'$
$W \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} W'$

The result follows by induction (using subject reduction and generation for the kinding hypothesis) and R-App:

$$\frac{S' \longrightarrow^{\star}_{\beta\top} (Fun(A{:}K)U_1')\ W' \qquad T \longrightarrow^{\star}_{\beta\top} V\ W \longrightarrow^{\star}_{\beta\top} V\ W' \qquad \Gamma' \vdash_{\mathcal{C}} Fun(A{:}K)U_1' \leq V}{\Gamma' \vdash_{\mathcal{C}} S' \leq T}$$

Subsubcase: $X = [W'/A]U_1'$
$U_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} U_1'$
$W \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} W'$

Then[7]

$$\begin{array}{lll}
\Gamma \vdash_{\mathcal{C}} Fun(A{:}K)U_1 \leq V & \Rightarrow & \text{(by induction)} \\
\Gamma' \vdash_{\mathcal{C}} Fun(A{:}K)U_1' \leq V & \Rightarrow & \text{(by R-App)} \\
\Gamma' \vdash_{\mathcal{C}} Fun(A{:}K)U_1'\ W' \leq V\ W' & \Rightarrow & \text{(by outer reduction on the left (6.4.7))} \\
\Gamma' \vdash_{\mathcal{C}} [W'/A]U_1' \leq V\ W' & \Rightarrow & \text{(by the expansion lemma (6.4.1))} \\
\Gamma' \vdash_{\mathcal{C}} S' \leq T, & &
\end{array}$$

using subject reduction and the generation lemma for the required kinding hypotheses in the first step and subject reduction in the third and fourth steps. □

6.4.9 Corollary [Reduction preserves subtyping]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $S \longrightarrow^{\star}_{\beta\top} S'$ and $T \longrightarrow^{\star}_{\beta\top} T'$, with $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma \vdash_{\mathcal{C}} S' \leq T'$.

[7] Here we can be more precise about why we choose parallel reduction to carry out this proof. A "commutation lemma" similar to the Church-Rosser property — the existence of a type S_1' for $S \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} S'$ and $S \longrightarrow^{\star}_{\beta\top} S_1$ where $S' \longrightarrow^{\star}_{\beta} S_1'$ and $S_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} S_1'$ (lemma 3.8) — is crucial for the induction. This immediately excludes the one-step reduction relation $\longrightarrow_{\beta\top}$. Ordinary many-step reduction, $\longrightarrow^{\star}_{\beta\top}$, is another obvious choice; but it cannot be used in the case of R-App, since here we need to know the *form* of the reduct of the application, which cannot be recovered from $\longrightarrow^{\star}_{\beta\top}$. Other, even more deterministic, reduction strategies such as normalizing reduction, leftmost-outermost reduction, or complete development might work here, but these all seem to fail in the R-Promote case. There, we have $S \longrightarrow^{\star}_{\beta\top} A\ S_1 \ldots S_n \uparrow_{\Gamma} \Gamma(A)\ S_1 \ldots S_n$, where $\Gamma(A)\ S_1$ can contain redices not present in $A\ S_1$. For the induction to work in this case, we need to be able to ignore these new redices when reducing further from $\Gamma(A)\ S_1 \ldots S_n$, which we would not be not free to do if we were using a more deterministic reduction strategy.

Proof: If we can do one $\longrightarrow\!\!\!\!\!\rightarrow_{\beta\top}$ step, we can do many. So the result follows by the observation (3.2) that the reflexive, transitive closure of parallel reduction coincides with ordinary many-step reduction. $\square$

6.4.10 Corollary [Cut-free derivations can be strengthened]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma \vdash_{\mathcal{CS}} S \leq T$.

Proof: By induction. $\square$

6.5 Completeness of Strong, Cut-free Subtyping

6.5.1 Theorem: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{R}} S \leq T$, then $\Gamma \vdash_{\mathcal{CS}} S \leq T$.

Proof: By induction on derivations. In all the cases except R-TRANS, we argue as follows: By the induction hypothesis (using the well-kindedness of subderivations), we may assume that all the subtyping premises are proved by cut-free derivations. Since the conclusion is not a cut, the whole derivation is cut-free and can be strengthened using Corollary 6.4.10.

Now, suppose the final rule is R-TRANS:

$$\frac{\Gamma \vdash_{\mathcal{R}} S \leq U \qquad \Gamma \vdash_{\mathcal{R}} U \leq T \qquad U \in K}{\Gamma \vdash_{\mathcal{R}} S \leq T}$$

By induction, we may assume that the derivations of the premises are cut-free. Moreover, by Corollary 6.4.9, we can put U in normal form: $\Gamma \vdash_{\mathcal{C}} S \leq U^!$ and $\Gamma \vdash_{\mathcal{C}} U^! \leq T$. Corollary 6.4.10 allows these derivations to be strengthened: $\Gamma \vdash_{\mathcal{CS}} S \leq U^!$ and $\Gamma \vdash_{\mathcal{CS}} U^! \leq T$. The result now follows from Proposition 6.3.2. $\square$

6.6 The Algorithm

We now show that the algorithm we developed informally in Section 6.1 is indeed a decision procedure for the subtype relation.

The first thing we must verify is that this recursively defined procedure is really an algorithm — that it halts in finite time on all well-kinded inputs.

6.6.1 Proposition [Termination of the algorithm]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. Then *check* halts when presented with $\Gamma \vdash S \leq T$ as input.

Proof: We use the fact that the $\beta\top\Gamma$-reduction defined in Section 5 is strongly normalizing to define a simple termination ordering for the algorithm.

First, note that the recursive call in the third clause (the clause for promotion) is guaranteed to halt on the next step if U, the promotion of S, reduces to $Top(K)$ for some K. Thus, we need only consider the possibility of nontermination in the case where the promotion of S is different from Top — i.e., where the variable being promoted is a Γ-redex.

Let the *rank* of a well-kinded type V in a context Γ be the pair (r, s), where r is the maximum length of a $\beta\top\Gamma$-reduction sequence starting from V and s is the number of characters in V. Let the rank of a well-kinded subtyping statement $\Gamma \vdash S \leq T$ the pairwise sum of the ranks of S and T. Order ranks lexicographically. Then the rank of every recursive call of *check* in its definition is smaller than the rank of the input (using the observation that we need only consider "interesting promotions" in the third clause). $\square$

The algorithm *check* does not include a case corresponding to the pointwise application rule R-APP. But this rule can easily be shown to be inessential in strong, cut-free derivations.

6.6.2 Lemma [Eliminability of R-APP]: Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$. If $\Gamma \vdash_{\mathcal{CS}} S \leq T$, then this statement can also be proved by a (strong, cut-free) derivation with no instances of R-APP.

Proof: Straightforward induction on derivations, using Lemma 6.3.1 for the R-APP case. In effect, each instance of the application rule is replaced by a sequence of instances of the promotion rule. $\square$

Finally, we verify that the algorithm defines the same relation as the original subtyping rules.

6.6.3 Theorem: The algorithm *check* is sound and complete for the original subtyping relation (on well-kinded types).

Proof: On well-kinded inputs, the original subtyping relation is equivalent to the reducing subtyping relation (6.2.7) restricted to strong, cut-free derivations (6.5.1) with no uses of R-App (6.6.2). Now, each of the rules in this restricted system begins by normalizing both sides of the conclusion. Nothing changes if we present the rules in a form where the conclusion is *assumed* to be in normal form, adding a single normalization step at the very end of the derivation and inserting a re-normalizing step at each premise that is not guaranteed to be in normal form when the conclusion is; indeed, there is just one of these: the last premise of R-Promote. We may now observe that proof search for derivations in the reformulated system is an essentially linear process: any given subtyping statement can match the conclusion of only one subtyping rule for which further search may be required. That is, a given statement may match R-Refl and/or R-Top and/or *one* of the remaining rules R-Promote, R-Arrow, R-All, and R-App. Since the premises of R-Refl and R-Top can be checked directly, the applicability of these two rules can be tested first. Using this strategy, no backtracking is required. Moreover, all the metavariables appearing in the premises the rules may be calculated from the conclusion: no guessing is required. The algorithm *check* implements this strategy. □

7 Typing

In Section 6, we derived an algorithm for checking the subtyping relation by controlling the non-syntax-directed rules of transitivity and conversion. In this section, we carry out an analogous exercise for the typing relation, eliminating the rule of subsumption from the system defined in Section 2.5 and accounting for its effects by extending some of the other rules.

Compared to what we had to do for subtyping, this is actually a rather simple task. Indeed, the resulting algorithm strongly resembles standard algorithms for typechecking $F_{\leq}$. The only essential difference comes from the fact that the promotion relation here must deal with application in addition to the promotion of type variables. As usual, we obtain the algorithm by analyzing the shapes of minimal types.

First, we check that the typing relation guarantees well-kindedness of derivable statements:

7.1 Lemma: If $\Gamma \vdash t \in T$, then $\Gamma \vdash T \in \star$.

The minimal type of an expression is a type smaller or equal to all the other types of the expression. For the algorithm, we also need to talk about a term's minimal types of certain specific shapes.

7.2 Definition [Minimal, arrow-minimal, and All-minimal types]:

1. A type S is *minimal* for a term s in a context Γ if $\Gamma \vdash s \in S$ and, for all T with $\Gamma \vdash s \in T$, we have $\Gamma \vdash S \leq T$.

2. A type $S_1 \rightarrow S_2$ is *arrow-minimal* for s in Γ if $\Gamma \vdash s \in S_1 \rightarrow S_2$ and, for all arrow-types $T_1 \rightarrow T_2$ with $\Gamma \vdash s \in T_1 \rightarrow T_2$, we have $\Gamma \vdash S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2$.

3. A type $All(A{\leq}S_1)S_2$ is *All-minimal* for s in Γ if $\Gamma \vdash s \in All(A{\leq}S_1)S_2$ and, for all All-types $All(A{\leq}T_1)\,T_2$ with $\Gamma \vdash s \in All(A{\leq}T_1)\,T_2$, we have $\Gamma \vdash All(A{\leq}S_1)\,S_2 \leq All(A{\leq}T_1)\,T_2$.

7.3 Definition: Let T be well kinded in Γ. The *promote-normal form* of T in Γ is

$$\Uparrow^{!}_{\Gamma} T = \begin{cases} \Uparrow^{!}_{\Gamma} U & \text{if } T^{!} \uparrow_{\Gamma} U\text{; or} \\ T^{!} & \text{if } T^{!} \text{ cannot be promoted.} \end{cases}$$

We next show how All-minimal and arrow-minimal types of a term can be calculated from its minimal type.

7.4 Lemma:

1. Suppose S and $T_1 \rightarrow T_2$ are well kinded. If $\Gamma \vdash S \leq T_1 \rightarrow T_2$, then $\Uparrow_\Gamma S = S_1 \rightarrow S_2$ and $\Gamma \vdash S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2$.

2. Suppose S and $All(A{\leq}T_1)\,T_2$ are well kinded. If $\Gamma \vdash S \leq All(A{\leq}T_1)\,T_2$, then $\Uparrow_\Gamma S = All(A{\leq}S_1)\,S_2$ and $\Gamma \vdash All(A{\leq}S_1)\,S_2 \leq All(A{\leq}T_1)\,T_2$.

Proof:

1. On well-kinded inputs, the original subtyping relation is equivalent to the reducing subtyping relation (6.2.7) restricted to strong, cut-free derivations (6.5.1). Proceed by induction on a derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T_1 \rightarrow T_2$.

2. Similar. □

7.5 Corollary:

1. If S is minimal for s in Γ and $\Uparrow_\Gamma S$ has the form $S_1 \rightarrow S_2$, then $S_1 \rightarrow S_2$ is arrow-minimal for s in Γ.

2. If S is minimal for s in Γ and $\Uparrow_\Gamma S$ has the form $All(A{\leq}S_1)\,S_2$, then $All(A{\leq}S_1)\,S_2$ is All-minimal for s in Γ.

The typing algorithm can now be obtained directly from the original typing relation by removing T-Subsumption — in effect, restricting the set of types derivable for a well-typed term to one of its minimal types — and generalizing the application and type application rules to compensate for this restriction in their premises. We use $\vdash_{\mathcal{A}}$ to distinguish the typing algorithm from the original typing relation.

7.6 Definition [Typechecking algorithm]:

$$\frac{\vdash \Gamma\ ok}{\Gamma \vdash_{\mathcal{A}} x \in \Gamma(x)} \qquad \text{(A-Var)}$$

$$\frac{\Gamma, x{:}T_1 \vdash_{\mathcal{A}} e \in T_2}{\Gamma \vdash_{\mathcal{A}} fun\,(x{:}T_1)\,e \in T_1 \rightarrow T_2} \qquad \text{(A-Arrow-I)}$$

$$\frac{\begin{array}{c}\Uparrow_\Gamma S = T_1 \rightarrow T_2 \\ \Gamma \vdash_{\mathcal{A}} s \in S \qquad \Gamma \vdash_{\mathcal{A}} t \in T \qquad \Gamma \vdash T \leq T_1\end{array}}{\Gamma \vdash_{\mathcal{A}} s\,t \in T_2} \qquad \text{(A-Arrow-E)}$$

$$\frac{\Gamma, A{\leq}T_1 \vdash_{\mathcal{A}} e \in T_2}{\Gamma \vdash_{\mathcal{A}} fun\,(A{\leq}T_1)\,e \in All(A{\leq}T_1)\,T_2} \qquad \text{(A-All-I)}$$

$$\frac{\begin{array}{c}\Uparrow_\Gamma T = All(A{\leq}T_1)\ T_2 \\ \Gamma \vdash_{\mathcal{A}} t \in T \qquad \Gamma \vdash S \in K \qquad \Gamma \vdash S \leq T_1\end{array}}{\Gamma \vdash_{\mathcal{A}} t\,S \in [S/A]T_2} \qquad \text{(A-All-E)}$$

The termination of this algorithm is straightforward, given the decidability of kinding, the termination of the subroutine for checking subtyping, and the strong normalization of $\beta\top\Gamma$-reduction, which guarantees that $\Uparrow_\Gamma T$ can be calculated in finite time whenever T is well kinded.

7.7 Fact:

1a. If $\vdash \Gamma\ ok$, then $\Gamma(x)$ is a minimal type of x in Γ.

1b. If $\not\vdash \Gamma\ ok$, then x has no type in Γ.

2a. If T_2 is a minimal type of e in $\Gamma, x{:}T_1$, then $T_1 \rightarrow T_2$ is a minimal type of $fun\,(x{:}T_1)\,e$ in Γ.

2b. If e has no type in $\Gamma, x{:}T_1$, then $fun\,(x{:}T_1)\,e$ has no type in Γ.

3a. If S is a minimal type for s in Γ and $\Uparrow^!_\Gamma S = T_1 \rightarrow T_2$ and T is a minimal type for t in Γ and $\Gamma \vdash T \leq T_1$, then T_2 is a minimal type for $s\ t$ in Γ.

3b. If s or t has no type in Γ, or if S is a minimal type for s in Γ and T is a minimal type for t in Γ but $\Uparrow^!_\Gamma S \neq T_1 \rightarrow T_2$, or if $\Uparrow^!_\Gamma S = T_1 \rightarrow T_2$ but $\Gamma \not\vdash T \leq T_1$, then $s\ t$ has no type in Γ.

4a. If T_2 is a minimal type of e in Γ, $A{\leq}T_1$, then $All(A{\leq}T_1)\,T_2$ is a minimal type of $fun\,(A{\leq}T_1)\,e$ in Γ.

4b. If e has no type in Γ, $A{\leq}T_1$, then $fun\,(A{\leq}T_1)\,e$ has no type in Γ.

5a. If S is a minimal type for s in Γ and $\Uparrow^!_\Gamma S = All(A{\leq}T_1)\,T_2$ and $\Gamma \vdash U \leq T_1$, then $[U/A]T_2$ is a minimal type for $s\ U$ in Γ.

5b. If s has no type in Γ or if S is a minimal type for s in Γ but $\Uparrow^!_\Gamma S \neq All(A{\leq}T_1)\,T_2$ or if $\Uparrow^!_\Gamma S = All(A{\leq}T_1)\,T_2$ but $\Gamma \not\vdash U \leq T_1$, then $s\ U$ has no type in Γ.

7.8 Theorem [Soundness and completeness]:

1. If $\Gamma \vdash_{\mathcal{A}} t \in T$ then $\Gamma \vdash T \in \star$ and $\Gamma \vdash t \in T$.
2. If $\Gamma \vdash s \in T$ then $\Gamma \vdash T \in \star$ and $\Gamma \vdash_{\mathcal{A}} s \in S$, where S is minimal for s in Γ.

Proof: By induction, using the previous facts. □

This brings us to our final result:

7.9 Corollary [Decidability of $F^\omega_\leq$ typing]: The original typing relation $\Gamma \vdash t \in T$ is decidable.

Proof: To check whether a statement $\Gamma \vdash t \in T$ is derivable, first check that T is well kinded, then calculate the minimal type S of t in Γ using the algorithm above and use the subtyping algorithm *check* to verify that $\Gamma \vdash S \leq T$. □

Acknowledgements

This research was supported by the *Specification and Verification of Distributed Systems* project, funded by the Deutsche Forschungsgemeinschaft, Sonderforschungsbereich 182 "Multiprozessor- und Netzwerkkonfigurationen" with assistance from the DAAD (Deutscher Akademischer Austauschdienst) [Steffen], by the British Science and Engineering Research Council [Pierce], and by and the British Council [Steffen and Pierce]. Discussions with Ulf Becker, Luca Cardelli, Adriana Compagnoni, Randy Pollack, Terry Stroup, and David Turner helped deepen our understanding of $F^\omega_\leq$. David Aspinall and Adriana Compagnoni gave us useful comments on a late draft. Giorgio Ghelli pointed out a bug in an earlier version of the proof, which Adriana Compagnoni helped correct.

References

[Aba93] Martín Abadi. Baby Modula-3 and a theory of objects. Research Report 95, Digital Equipment Corporation, Systems Research Center, Palo Alto, California, February 1993. To appear in *Journal of Functional Programming*.

[AC94a] Martín Abadi and Luca Cardelli. A theory of primitive objects: Second-order systems. In *European Symposium on Programming (ESOP), Edinburgh, Scotland*, 1994.

[AC94b] Martín Abadi and Luca Cardelli. A theory of primitive objects: Untyped and first-order systems. In *Theoretical Aspects of Computer Software (TACS), Sendai, Japan*, 1994.

[Bar84] Hendrik Barendregt. *The Lambda Calculus: Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, revised edition, 1984.

[BCGS91] Val Breazu-Tannen, Thierry Coquand, Carl Gunter, and Andre Scedrov. Inheritance as implicit coercion. *Information and Computation*, 93:172–221, 1991.

[BL90] Kim B. Bruce and Giuseppe Longo. A modest model of records, inheritance, and bounded quantification. *Information and Computation*, 87:196–240, 1990. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press; to appear, 1994). An earlier version appeared in the proceedings of the IEEE Symposium on Logic in Computer Science, 1988.

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

[Bru93] Kim B. Bruce. Safe type checking in a statically typed object-oriented programming language. In *Proceedings of the Twentieth ACM Symposium on Principles of Programming Languages*, January 1993. To appear in *Journal of Functional Programming*.

[Car84] Luca Cardelli. A semantics of multiple inheritance. In G. Kahn, D. MacQueen, and G. Plotkin, editors, *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*, pages 51–67. Springer-Verlag, 1984. Full version in *Information and Computation* 76:138–164, 1988.

[Car88] Luca Cardelli. Structural subtyping and the notion of power type. In *Proceedings of the 15th ACM Symposium on Principles of Programming Languages*, pages 70–79, San Diego, CA, January 1988.

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

[Car92] Luca Cardelli. Extensible records in a pure calculus of subtyping. Research report 81, DEC Systems Research Center, January 1992. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press; to appear, 1994).

[Cas92] Giuseppe Castagna. Strong typing in object-oriented paradigms. Rapport de Recherche LIENS-92-11, Ecole Normale Supérieure, Paris, May 1992.

[CCH+89] Peter Canning, William Cook, Walter Hill, Walter Olthoff, and John Mitchell. F-bounded quantification for object-oriented programming. In *Fourth International Conference on Functional Programming Languages and Computer Architecture*, pages 273–280, September 1989.

[CG91] Pierre-Louis Curien and Giorgio Ghelli. Subtyping + extensionality: Confluence of $\beta\eta$-reductions in $F_{\leq}$. In Ito and Meyer [IM91], pages 731–749.

[CG92] Pierre-Louis Curien and Giorgio Ghelli. Coherence of subsumption: Minimum typing and type-checking in $\mathrm{F}_{\leq}$. *Mathematical Structures in Computer Science*, 2:55–91, 1992. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press; to appear, 1994).

[CGL92] G. Castagna, G. Ghelli, and G. Longo. A calculus for overloaded functions with subtyping. In *ACM conference on LISP and Functional Programming*, pages 182–192, San Francisco, July 1992. ACM Press. Also available as Rapport de Recherche LIENS-92-4, Ecole Normale Supérieure, Paris.

[CHC90] William R. Cook, Walter L. Hill, and Peter S. Canning. Inheritance is not subtyping. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 125–135, San Francisco, CA, January 1990. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press; to appear, 1994).

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5:56–68, 1940.

[CL91] Luca Cardelli and Giuseppe Longo. A semantic basis for Quest. *Journal of Functional Programming*, 1(4):417–458, October 1991. Preliminary version in ACM Conference on Lisp and Functional Programming, June 1990. Also available as DEC SRC Research Report 55, Feb. 1990.

[CMMS91] Luca Cardelli, Simone Martini, John C. Mitchell, and Andre Scedrov. An extension of system F with subtyping. In Ito and Meyer [IM91], pages 750–770.

[Com94] Adriana B. Compagnoni. Subtyping in $F^{\omega}_{\wedge}$ is decidable. Draft technical report, January 1994.

[CP93] Adriana B. Compagnoni and Benjamin C. Pierce. Multiple inheritance via intersection types. Technical Report ECS-LFCS-93-275, LFCS, University of Edinburgh, August 1993. Also available as Catholic University Nijmegen computer science technical report 93-18.

[CP94] Giuseppe Castagna and Benjamin Pierce. Decidable bounded quantification. In *Proceedings of Twenty-First Annual ACM Symposium on Principles of Programming Languages, Portland, Oregon*. ACM, January 1994.

[CW85] Luca Cardelli and Peter Wegner. On understanding types, data abstraction, and polymorphism. *Computing Surveys*, 17(4), December 1985.

[Gal90] Jean H. Gallier. On Girard's "candidats de reductibilité". In Piergiorgio Odifreddi, editor, *Logic and Computer Science*, number 31 in APIC Studies in Data Processing, pages 123–203. Academic Press, 1990.

[Ghe90] Giorgio Ghelli. *Proof Theoretic Studies about a Minimal Type System Integrating Inclusion and Parametric Polymorphism.* PhD thesis, Università di Pisa, March 1990. Technical report TD–6/90, Dipartimento di Informatica, Università di Pisa.

[Ghe93a] G. Ghelli. Divergence of $F_{\leq}$ type checking. Technical Report 5/93, Dipartimento di Informatica, Università di Pisa, 1993.

[Ghe93b] Giorgio Ghelli. Recursive types are not conservative over $F_{\leq}$. In *Typed Lambda Calculus and Applications*, March 1993.

[Gir72] Jean-Yves Girard. *Interprétation fonctionelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, Université Paris VII, 1972.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science.* Cambridge University Press, Cambridge, 1989.

[GP92] Giorgio Ghelli and Benjamin Pierce. Bounded existentials and minimal typing. Draft report, June 1992.

[HP94] Martin Hofmann and Benjamin Pierce. A unifying type-theoretic framework for objects. In *Symposium on Theoretical Aspects of Computer Science*, 1994. Extended version available as "An Abstract View of Objects and Subtyping (Preliminary Report)," University of Edinburgh, LFCS technical report ECS-LFCS-92-226, 1992.

[IM91] T. Ito and A. R. Meyer, editors. *Theoretical Aspects of Computer Software (Sendai, Japan)*, number 526 in Lecture Notes in Computer Science. Springer-Verlag, September 1991.

[KS92] Dinesh Katiyar and Sriram Sankar. Completely bounded quantification is decidable. In *Proceedings of the ACM SIGPLAN Workshop on ML and its Applications*, June 1992.

[Mar88] Simone Martini. Bounded quantifiers have interval models. In *Proceedings of the ACM Conference on Lisp and Functional Programming*, pages 174–183, Snowbird, Utah, July 1988. ACM.

[MHF93] John C. Mitchell, Furio Honsell, and Kathleen Fisher. A lambda calculus of objects and method specialization. In *1993 IEEE Symposium on Logic in Computer Science*, June 1993.

[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. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press; to appear, 1994).

[Pie92] Benjamin C. Pierce. Bounded quantification is undecidable. In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Programming Languages, (Albequerque, New Mexico)*, January 1992. To appear in *Information and Computation*, and in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press; to appear, 1994).

[PT93a] Benjamin C. Pierce and David N. Turner. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming*, 1993. To appear; a preliminary version appeared in Principles of Programming Languages, 1993, and as University of Edinburgh technical report ECS-LFCS-92-225, under the title "Object-Oriented Programming Without Recursive Types".

[PT93b] Benjamin C. Pierce and David N. Turner. Statically typed friendly functions via partially abstract types. Technical Report ECS-LFCS-93-256, University of Edinburgh, LFCS, April 1993. Also available as INRIA-Rocquencourt Rapport de Recherche No. 1899.

[Rey74] John Reynolds. Towards a theory of type structure. In *Proc. Colloque sur la Programmation*, pages 408–425, New York, 1974. Springer-Verlag LNCS 19.

[SP93] Paula Severi and Erik Poll. Pure type systems with definitions. Computing science note 93/24, Eindhoven University of Technology, September 1993.

[Wan87] Mitchell Wand. Complete type inference for simple objects. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, Ithaca, NY, June 1987.

[Wan88] Mitchell Wand. Corrigendum: Complete type inference for simple objects. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, 1988.

[Wan89] Mitchell Wand. Type inference for record concatenation and multiple inheritance. In *Fourth Annual IEEE Symposium on Logic in Computer Science*, pages 92–97, Pacific Grove, CA, June 1989.