Bounded Quantification with Bottom

Benjamin C. Pierce

Computer Science Department
Indiana University
Lindley Hall 215
Bloomington, IN 47405, USA
pierce@cs.indiana.edu

Indiana University
CSCI Technical Report #492

November 12, 1997

Abstract

While numerous extensions of Cardelli and Wegner's calculus of polymorphism and subtyping, Kernel Fun, have been studied during the past decade, one quite simple one has received little attention: the addition of a minimal type Bot, dual to the familiar maximal type Top. We develop basic meta-theory for this extension. Although most of the usual properties of Kernel Fun (existence of meets and joins, decidability of subtyping and typing, subject reduction, etc.) also hold for the extended system, the presence of Bot introduces some surprising intricacies. In particular, a type variable bounded by Bot is actually a synonym for Bot; such "bottom variables" must be treated specially at several points.

1 Introduction

The typed lambda-calculus Kernel Fun [CW85] and its wilder sibling, $F_\leq$ [CG92, Ghe90, CMMS94] have become standard tools for the foundational study of programming languages combining subtyping and impredicative polymorphism. Their syntax and semantics have been investigated in detail, and they have formed the basis for a number of experimental language designs. Moreover, many ways of enriching these pure systems have been studied, including extensions with recursive types [AC93], extensible records [CM91], existential types [GP97], and type operators [Car90, CL91, PT94, HP95, PS94, Com94].

However, one quite obvious extension has apparently never been considered in detail. Most presentations of Kernel Fun (and $F_\leq$) include a type Top that is used, among other things, to recover ordinary unbounded quantification from bounded quantification, achieving the effect of All(X)T by writing All(X<:Top)T instead. The Top type is axiomatized as a maximal element in the subtype relation, by including a subtyping rule

$$\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{Top} \qquad \text{(S-Top)}$$

that makes Top a supertype of T for every type T. It is natural to ask whether it also makes sense to add a type Bot with the dual subtyping rule:

$$\Gamma \vdash \mathtt{Bot} \mathrel{<:} \mathtt{T} \qquad \text{(S-Bot)}$$

The potential applications for such an extension are numerous. For example:

1. In a language with exceptions, a natural type for the `raise` construct is

   `raise` $\in$ `exception` $\rightarrow$ `Bot`.

   Intuitively, `Bot` here is the type of computations that do not return an answer.

2. `Bot` is useful in typing the "leaf nodes" of polymorphic data structures. For example, `List(Bot)` is a good type for `nil`.

3. `Bot` is a natural type for the "uninitialized pointer" value of languages like Java: saying that "`null` has every type" is just the same as "`null` $\in$ `Bot`."[1]

4. A type system including both `Top` and `Bot` seems to be a natural target for *type inference*, allowing the constraints on an omitted type parameter to be captured by a pair of bounds: we write `S<:X<:T` to mean "the value of `X` must lie somewhere between `S` and `T`." In such a scheme, a completely unconstrained parameter is bounded below by `Bot` and above by `Top`.

   (This application to type inference was the one that actually motivated the present study. It is described in detail in [PT98, PT97].)

Unfortunately, the addition of `Bot` significantly complicates the meta-theoretic properties of the system. Most significantly, in a type of the form `All(X<:Bot)T`, the variable `X` is actually a *synonym* for `Bot` inside `T`, since `X` is a subtype of `Bot` by assumption and `Bot` is a subtype of `X` by the rule S-Bot. This, in turn, means that pairs of types such as

`All(X<:Bot)X→X`

and

`All(X<:Bot)Bot→Bot`

will be equivalent in the subtype relation, even though they are not syntactically identical. Worse yet, if the ambient context contains the assumptions `X<:Bot` and `Y<:Bot`, then the types

`X→Y`

and

`Y→X`

are equivalent even though neither of them mentions `Bot` explicitly!

Despite these difficulties, the essential properties of Kernel Fun can also be established in the presence of `Bot`. The main ones developed here are:

1. "Cut-elimination" (admissibility of transitivity) for subtyping.

2. Existence of meets and joins in the subtype relation.

   Because the existence of meets and joins in the subtype relation is crucial for the development here and in [PT97], the sytem studied here extends Cardelli and Wegner's original Kernel Fun [CW85] rather than the "Full $F_{\leq}$" of Curien and Ghelli [CG92]. The only difference between these systems lies in the subtyping rule for comparing quantified types. In Kernel Fun, the bounds of the two types must be the same; in Full $F_{\leq}$, the subtype relation is contravariant in the bounds. Although the additional power offered by Full $F_{\leq}$ is occasionally useful (e.g. [ACV96]), the system lacks many desirable properties, such as decidability [Pie94].

3. Completeness of subtyping and typechecking algorithms.

4. Subject reduction.

[1] Strictly speaking, Java's `null` value has all *object* types, but not types like integer or array. The languge studied here does not capture this refinement.

5. Soundness and completeness of an algorithm for calculating the least supertype of a given type in which some variable does not occur free.

The principal points of difference from ordinary Kernel Fun are:

1. The algorithm for calculating meets and joins in the subtype relation is slightly different (mainly because in pure Kernel Fun not every pair of types has a common lower bound, while here `Bot` lies below all types). The completeness proof for the algorithm requires a more refined argument because `Bot` introduces some extra possibilities.

2. Some work is needed to characterize the circumstances under which the equivalence class of a type in the subtype relation will be a singleton. (In pure Kernel Fun, all equivalence classes are singletons.)

3. In the definition of the algorithmic typing relation, extra rules are needed to handle the cases where the function part of an application or the guard of a `case` has type `Bot`. These cases must later be treated in the proof of subject reduction.

4. The algorithm for finding the smallest supertype of a given type that does not use a given type variable is actually somewhat simpler than the corresponding algorithm for Kernel Fun [GP97].

The following section introduces the syntax of the extended system formally. The remainder of the paper is broken into two main parts: Sections 3 and 4 develop the properties of subtyping, while Section 5 defines the typing relation and proves subject reduction.

2 Syntax

Besides the addition of `Bot`, Kernel Fun is extended in two straightforward ways for purposes of the present study. First, we include a type constructor `List` with polymorphic constructors `nil` and `cons` and a `case` construct. These constructs illustrate the use of joins in the treatment of many constructs found in larger programming languages. Second, type and term abstractions are combined into one syntactic construct and several arguments (types and terms) can be passed at the same time. This generalization has no effect on the properties of the system *per se*, but plays an important role in the scheme for inferring type arguments described in [PT97]; it is used here for notational consistency with that paper.

The syntax of types, terms, and typing contexts is as follows:

$\texttt{T}$	$::=$	$\texttt{Top}$	maximal type
		$\texttt{Bot}$	minimal type
		$\texttt{X}$	type variable
		$\texttt{All}(\overline{\texttt{X}}\texttt{<:}\overline{\texttt{T}})\overline{\texttt{T}}\rightarrow\texttt{T}$	(polymorphic) function type
		$\texttt{List(T)}$	list type
$\texttt{e}$	$::=$	$\texttt{c}$	constant
		$\texttt{x}$	variable
		$\texttt{fun}[\overline{\texttt{X}}\texttt{<:}\overline{\texttt{T}}](\overline{\texttt{x}}\texttt{:}\overline{\texttt{T}})\texttt{e}$	abstraction
		$\texttt{e}[\overline{\texttt{T}}](\overline{\texttt{e}})$	application
		$\texttt{case e}_1\ \texttt{of nil}\rightarrow\texttt{e}_2\ \texttt{\|}\ \texttt{cons(x,y)}\rightarrow\texttt{e}_3$	list case
Γ	$::=$	$\bullet$	empty context
		$\Gamma, \texttt{x:T}$	variable binding
		$\Gamma, \texttt{X<:T}$	type variable binding

Throughout the paper, we use overbars to denote finite sequences: $\overline{\texttt{X}}$ denotes a sequence $\texttt{X}_1,\ldots,\texttt{X}_n$ of type variables, $\overline{\texttt{x}}\texttt{:}\overline{\texttt{T}}$ denotes a sequence of bindings $\texttt{x}_1\texttt{:T}_1,\ldots,\texttt{x}_n\texttt{:T}_n$, etc. Our constants `c` include the list constants `nil` and `cons`, which have type `All(X<:Top)List(X)` and `All(X<:Top)(X,List(X))`$\rightarrow$`List(X)` respectively.[2]

Note that the concrete syntax for multi-abstraction and application is designed for brevity of the present theoretical development: in an actual programming language, it might be better to choose something more verbose and easier to read. Also, note that ordinary functions and regular polymorphic functions can be obtained from our combined syntax by using a zero-length sequence of either type or value parameters. We write $\overline{\mathtt{S}}\rightarrow\mathtt{T}$ as an abbreviation for the monomorphic function type $\mathtt{All()}\overline{\mathtt{S}}\rightarrow\mathtt{T}$. Similarly, we write $\mathtt{fun(}\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{T}}\mathtt{)e}$ as an abbreviation for the monomorphic function $\mathtt{fun[](}\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{T}}\mathtt{)e}$.

Types, terms, contexts, and judgements that differ only in the names of bound variables are regarded as identical. Binders in contexts are assumed to have distinct names; when a new binding is added to a context, it is assumed to have been renamed so as to maintain this invariant. The rules for scoping of bound variables are as usual; in particular, in $\mathtt{All(}\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{S}}\mathtt{)}\overline{\mathtt{T}}\rightarrow\mathtt{U}$ the scope of each $\mathtt{X}_i$ is everything to its right—the bounds $\mathtt{S}_{i+1}$ through $\mathtt{S}_n$, all of the $\mathtt{T}_j$s, and $\mathtt{U}$. The set of type variables free in $\mathtt{T}$, written $FV(\mathtt{T})$, is defined as usual.

All typing contexts are assumed to be "well scoped," in the sense that all the free variables in each binding appear in the domain of the context further to the left.

Write $\Gamma(\mathtt{X})$ for the bound of $\mathtt{X}$ in Γ and $\Gamma(\mathtt{x})$ for the type of $\mathtt{x}$ in Γ.

3 Subtyping

The subtyping relation is the obvious extension of the Kernel Fun subtyping relation with `List` and `Bot` types. We write $\Gamma \vdash \overline{\mathtt{S}} \mathrel{\mathtt{<:}} \overline{\mathtt{T}}$ to mean "$|\overline{\mathtt{S}}| = |\overline{\mathtt{T}}|$ and $\Gamma \vdash \mathtt{S}_i \mathrel{\mathtt{<:}} \mathtt{T}_i$ for all $1 \le i \le |\mathtt{S}|$."

$$\Gamma \vdash \mathtt{T} \mathrel{\mathtt{<:}} \mathtt{Top} \tag{S-Top}$$

$$\Gamma \vdash \mathtt{Bot} \mathrel{\mathtt{<:}} \mathtt{T} \tag{S-Bot}$$

$$\Gamma \vdash \mathtt{X} \mathrel{\mathtt{<:}} \mathtt{X} \tag{S-Refl}$$

$$\frac{\Gamma \vdash \Gamma(\mathtt{X}) \mathrel{\mathtt{<:}} \mathtt{T}}{\Gamma \vdash \mathtt{X} \mathrel{\mathtt{<:}} \mathtt{T}} \tag{S-Var}$$

$$\frac{\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{B}} \vdash \overline{\mathtt{T}} \mathrel{\mathtt{<:}} \overline{\mathtt{S}} \qquad \Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{B}} \vdash \mathtt{Q} \mathrel{\mathtt{<:}} \mathtt{R}}{\Gamma \vdash \mathtt{All(}\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{B}}\mathtt{)}\overline{\mathtt{S}}\rightarrow\mathtt{Q} \mathrel{\mathtt{<:}} \mathtt{All(}\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{B}}\mathtt{)}\overline{\mathtt{T}}\rightarrow\mathtt{R}} \tag{S-Fun}$$

$$\frac{\Gamma \vdash \mathtt{S} \mathrel{\mathtt{<:}} \mathtt{T}}{\Gamma \vdash \mathtt{List(S)} \mathrel{\mathtt{<:}} \mathtt{List(T)}} \tag{S-List}$$

Two points are worth noting. First, as discussed in the introduction, the definition extends the original "Kernel Fun variant" of $F_{\le}$ [CW85], in which the upper bounds $\overline{\mathtt{B}}$ in the subtyping rule for polymorphic functions are required to be identical, rather than the more powerful but less tractable variant of Curien and Ghelli [CG92, CMMS94]. The principal reason for this restriction is that it allows us to define meets and joins of all pairs of types, which may fail to exist in "Full $F_{\le}$" [Ghe90]. Second, we give an algorithmic presentation of subtyping, in which the rules of transitivity and general reflexivity are omitted, and recover these rules (in the next subsection) as properties of the definition.

[2] The type of `nil` is chosen here for consistency with standard treatments. As remarked in the introduction, we could alternatively give `nil` the type `List(Bot)`. Since `nil` and `cons` are both constants, such choices have little effect on the general properties of the system (the proof of subject reduction would need to change slightly).

3.1 Basic Properties

Let us now commence the theoretical study of Kernel Fun with Bot, starting with some simple properties of the subtyping relation.

3.1.1 Lemma [Weakening]: If $\Gamma \vdash \texttt{S} <: \texttt{T}$, then $\Gamma,\Delta \vdash \texttt{S} <: \texttt{T}$.

Proof: Straightforward induction on derivations. ∎

3.1.2 Lemma [Strengthening for term variables]: If $\Gamma, \texttt{x}:\texttt{V}, \Delta \vdash \texttt{S} <: \texttt{T}$, then $\Gamma, \Delta \vdash \texttt{S} <: \texttt{T}$.

Proof: Immediate. ∎

3.1.3 Lemma [Reflexivity]: $\Gamma \vdash \texttt{T} <: \texttt{T}$ for all types T.

Proof: Straightforward induction on derivations. ∎

3.1.4 Lemma [Transitivity]: If $\Gamma \vdash \texttt{S} <: \texttt{T}$ and $\Gamma \vdash \texttt{T} <: \texttt{U}$, then $\Gamma \vdash \texttt{S} <: \texttt{U}$.

Proof: By induction on the sum of the sizes of the two given derivations.
If the final rule in either of the given derivations is the axiom $\Gamma \vdash \texttt{X} <: \texttt{X}$, then the result is immediate. Otherwise, we proceed by cases on the final rule used in the derivation of $\Gamma \vdash \texttt{S} <: \texttt{T}$.

Case S-Var: $\texttt{S} = \texttt{X}$ and $\Gamma \vdash \Gamma(\texttt{X}) <: \texttt{T}$

The induction hypothesis gives us $\Gamma \vdash \Gamma(\texttt{X}) <: \texttt{U}$, from which the result follows by S-Var.

Case S-Top: $\texttt{T} = \texttt{Top}$

By inspecting the definition of subtyping we see that U can only be Top, from which the result is immediate.

Case S-Bot: $\texttt{S} = \texttt{Bot}$

Immediate.

Case S-Fun: $\texttt{S} = \texttt{All}(\overline{\texttt{X}}<:\overline{\texttt{A}})\overline{\texttt{B}}\rightarrow\texttt{C}$ and $\texttt{T} = \texttt{All}(\overline{\texttt{X}}<:\overline{\texttt{A}})\overline{\texttt{D}}\rightarrow\texttt{E}$, with $\Gamma, \overline{\texttt{X}}<:\overline{\texttt{A}} \vdash \overline{\texttt{D}} <: \overline{\texttt{B}}$ and $\Gamma, \overline{\texttt{X}}<:\overline{\texttt{A}} \vdash \texttt{C} <: \texttt{E}$

By the definition of subtyping, either $\texttt{U} = \texttt{Top}$ or else $\texttt{U} = \texttt{All}(\overline{\texttt{X}}<:\overline{\texttt{A}})\overline{\texttt{F}}\rightarrow\texttt{G}$, with $\Gamma, \overline{\texttt{X}}<:\overline{\texttt{A}} \vdash \overline{\texttt{F}} <: \overline{\texttt{D}}$ and $\Gamma, \overline{\texttt{X}}<:\overline{\texttt{A}} \vdash \texttt{E} <: \texttt{G}$. Two uses of the induction hypothesis yield $\Gamma, \overline{\texttt{X}}<:\overline{\texttt{A}} \vdash \overline{\texttt{F}} <: \overline{\texttt{B}}$ and $\Gamma, \overline{\texttt{X}}<:\overline{\texttt{A}} \vdash \texttt{C} <: \texttt{G}$, from which the desired result follows by the subtyping rule for functions.

Case S-List: $\texttt{S} = \texttt{List(A)}$ and $\texttt{T} = \texttt{List(B)}$, with $\Gamma \vdash \texttt{A} <: \texttt{B}$

Similar. ∎

Another technical property that will be important in many proofs shows how subtyping is preserved under substitution.

3.1.5 Lemma [Substitution preserves subtyping]: If $\Gamma, \texttt{X}<:\texttt{B}, \Delta \vdash \texttt{S} <: \texttt{T}$ and $\Gamma \vdash \texttt{R} <: \texttt{B}$, then $\Gamma, [\texttt{R/X}]\Delta \vdash [\texttt{R/X}]\texttt{S} <: [\texttt{R/X}]\texttt{T}$, where $[\texttt{R/X}]\Delta$ denotes the substitution of R for X in all the bindings in Δ.

Proof: By induction on the depth of the derivation of $\Gamma, \texttt{X}<:\texttt{B}, \Delta \vdash \texttt{S} <: \texttt{T}$.

Case S-Refl: $\texttt{S} = \texttt{Y}$ and $\texttt{T} = \texttt{Y}$

Immediate, since $\Gamma, [\texttt{R/X}]\Delta \vdash [\texttt{R/X}]\texttt{Y} <: [\texttt{R/X}]\texttt{Y}$ by reflexivity (3.1.3).

Case S-Var: $\texttt{S} = \texttt{Y}$ and $\Gamma, \texttt{X}<:\texttt{B}, \Delta \vdash (\Gamma, \texttt{X}<:\texttt{B}, \Delta)(\texttt{Y}) <: \texttt{T}$

There are three subcases to consider, depending on where Y is bound in the context.

First, suppose $\texttt{Y} \in \mathit{dom}(\Gamma)$. By the induction hypothesis, we have $\Gamma, [\texttt{R/X}]\Delta \vdash [\texttt{R/X}](\Gamma(\texttt{Y})) <: [\texttt{R/X}]\texttt{T}$. However, since $\Gamma(\texttt{Y})$ cannot mention X as a free variable, we have $[\texttt{R/X}]\Gamma(\texttt{Y}) = \Gamma(\texttt{Y})$, and the result follows, since $[\texttt{R/X}]\texttt{Y} = \texttt{Y}$.

Next, suppose $\texttt{Y} = \texttt{X}$. By the induction hypothesis, we have $\Gamma, [\texttt{R/X}]\Delta \vdash [\texttt{R/X}]\texttt{B} <: [\texttt{R/X}]\texttt{T}$. However, since B cannot mention X as a free variable, we have $[\texttt{R/X}]\texttt{B} = \texttt{B}$. Since $\Gamma \vdash \texttt{R} <: \texttt{B}$ we obtain that $\Gamma, [\texttt{R/X}]\Delta \vdash \texttt{R} <: \texttt{B}$ from weakening (Lemma 3.1.1), and the result follows by transitivity (3.1.4).

Finally, suppose $\texttt{Y} \in \mathit{dom}(\Delta)$. By the induction hypothesis, we have $\Gamma, [\texttt{R/X}]\Delta \vdash ([\texttt{R/X}]\Delta)(\texttt{Y}) <: [\texttt{R/X}]\texttt{T}$. We can therefore conclude that $\Gamma, [\texttt{R/X}]\Delta \vdash \texttt{Y} <: [\texttt{R/X}]\texttt{T}$ by S-Var, and the result follows since $[\texttt{R/X}]\texttt{Y} = \texttt{Y}$.

Case S-TOP: $\mathtt{T} = \mathtt{Top}$

Immediate, since $\Gamma,\ \mathtt{[R/X]}\Delta \vdash \mathtt{[R/X]S}$ <: `Top` by S-TOP.

Case S-BOT: $\mathtt{S} = \mathtt{Bot}$

Immediate, since $\Gamma,\ \mathtt{[R/X]}\Delta \vdash \mathtt{Bot}$ <: `[R/X]T` by S-BOT.

Case S-FUN:

$$\begin{array}{l} \mathtt{S} = \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{B}}\rightarrow\mathtt{C} \\ \mathtt{T} = \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{D}}\rightarrow\mathtt{E} \\ \Gamma,\ \mathtt{X<:B},\ \Delta,\ \overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}} \vdash \overline{\mathtt{D}}\ \mathtt{<:}\ \overline{\mathtt{B}} \\ \Gamma,\ \mathtt{X<:B},\ \Delta,\ \overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}} \vdash \mathtt{C}\ \mathtt{<:}\ \mathtt{E} \end{array}$$

By the induction hypothesis, we have $\Gamma,\ \mathtt{[R/X]}(\Delta,\ \overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}}) \vdash \mathtt{[R/X]}\overline{\mathtt{D}}\ \mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{B}}$ and $\Gamma,\ \mathtt{[R/X]}(\Delta,\ \overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}}) \vdash \mathtt{[R/X]C}\ \mathtt{<:}\ \mathtt{[R/X]E}$, that is, $\Gamma,\ \mathtt{[R/X]}\Delta,\ \overline{\mathtt{Y}}\mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{A}} \vdash \mathtt{[R/X]}\overline{\mathtt{D}}\ \mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{B}}$ and $\Gamma,\ \mathtt{[R/X]}\Delta,\ \overline{\mathtt{Y}}\mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{A}} \vdash \mathtt{[R/X]C}\ \mathtt{<:}\ \mathtt{[R/X]E}$. By S-FUN, $\Gamma,\ \mathtt{[R/X]}\Delta \vdash \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{A}})\,\mathtt{[R/X]}\overline{\mathtt{B}}\rightarrow\mathtt{[R/X]C}\ \mathtt{<:}\ \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{A}})\,\mathtt{[R/X]}\overline{\mathtt{D}}\rightarrow\mathtt{[R/X]E}$ and the result follows since

$$\begin{array}{lcl} \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{A}})\,\mathtt{[R/X]}\overline{\mathtt{B}}\rightarrow\mathtt{[R/X]C} & = & \mathtt{[R/X]}(\mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{B}}\rightarrow\mathtt{C}) \\ \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\ \mathtt{[R/X]}\overline{\mathtt{A}})\,\mathtt{[R/X]}\overline{\mathtt{D}}\rightarrow\mathtt{[R/X]E} & = & \mathtt{[R/X]}(\mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{D}}\rightarrow\mathtt{E}). \end{array}$$

Case S-LIST: $\mathtt{S} = \mathtt{List(P)}$ and $\mathtt{T} = \mathtt{List(Q)}$, with $\Gamma,\ \mathtt{X<:B},\ \Delta \vdash \mathtt{P}\ \mathtt{<:}\ \mathtt{Q}$

Using induction, we have $\Gamma,\ \mathtt{[R/X]}\Delta \vdash \mathtt{[R/X]P}\ \mathtt{<:}\ \mathtt{[R/X]Q}$ and the result follows by S-LIST. ∎

The last basic property records some simple observations about the structure of the subtype relation for use in later arguments.

3.1.6 Lemma: If $\Gamma \vdash \mathtt{S}\ \mathtt{<:}\ \mathtt{T}$, then one of the following must hold:

1. $\mathtt{S} = \mathtt{Bot}$; or
2. $\mathtt{S} = \mathtt{Y}$, with $\Gamma \vdash \Gamma(\mathtt{Y})\ \mathtt{<:}\ \mathtt{T}$ as a subderivation; or
3. The shape of `S` can be derived from the shape of `T`:
   (a) if $\mathtt{T} = \mathtt{X}$ then $\mathtt{S} = \mathtt{X}$;
   (b) if $\mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}})\overline{\mathtt{A}}\rightarrow\mathtt{B}$ then $\mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}})\overline{\mathtt{V}}\rightarrow\mathtt{P}$, with $\Gamma,\ \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{A}}\ \mathtt{<:}\ \overline{\mathtt{V}}$ and $\Gamma,\ \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}} \vdash \mathtt{B}\ \mathtt{<:}\ \mathtt{P}$ as subderivations;
   (c) if $\mathtt{T} = \mathtt{List(A)}$ then $\mathtt{S} = \mathtt{List(P)}$, with $\Gamma \vdash \mathtt{P}\ \mathtt{<:}\ \mathtt{A}$ as a subderivation.

Proof: By inspection of the definition of subtyping. ∎

3.2 Promotion

The definition of the typechecking algorithm in Section 5 will need to be able to calculate, for each type `T`, the least supertype of `T` that has the shape of a function or list type (if any). Formally, write $\Gamma \vdash \mathtt{S} \Uparrow \mathtt{T}$ to mean "`T` is the *least nonvariable supertype* of `S`," defined by repeated promotion of variables as follows:

$$\frac{\mathtt{S}\text{ is not a variable}}{\Gamma \vdash \mathtt{S} \Uparrow \mathtt{S}}$$

$$\frac{\Gamma \vdash \Gamma(\mathtt{X}) \Uparrow \mathtt{T}}{\Gamma \vdash \mathtt{X} \Uparrow \mathtt{T}}$$

It is easy to check that these rules define a total function. Moreover:

3.2.1 Lemma: Suppose $\Gamma \vdash \mathtt{S} \Uparrow \mathtt{T}$.

1. $\Gamma \vdash \mathtt{S}\ \mathtt{<:}\ \mathtt{T}$.
2. If $\Gamma \vdash \mathtt{S}\ \mathtt{<:}\ \mathtt{U}$ and `U` is not a variable, then $\Gamma \vdash \mathtt{T}\ \mathtt{<:}\ \mathtt{U}$.

Proof: Part (1) is easy. Part (2) goes by straightforward induction on a derivation of $\Gamma \vdash \mathtt{S} <: \mathtt{U}$ (the only inductive case is the promotion rule S-VAR). ∎

In the proofs in later sections, we will need a few facts about how subtyping and promotion interact; they are collected in the following lemma.

3.2.2 Lemma: Suppose $\Gamma \vdash \mathtt{R} <: \mathtt{S}$.

1. If $\Gamma \vdash \mathtt{S} \Uparrow \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{M}})\overline{\mathtt{N}}\rightarrow\mathtt{O}$, then either
   (a) $\Gamma \vdash \mathtt{R} \Uparrow \mathtt{Bot}$ or
   (b) $\Gamma \vdash \mathtt{R} \Uparrow \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{M}})\overline{\mathtt{P}}\rightarrow\mathtt{Q}$ with $\Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{M}} \vdash \overline{\mathtt{N}} <: \overline{\mathtt{P}}$ and $\Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{M}} \vdash \mathtt{Q} <: \mathtt{O}$.
2. If $\Gamma \vdash \mathtt{S} \Uparrow \mathtt{List(Q)}$, then either
   (a) $\Gamma \vdash \mathtt{R} \Uparrow \mathtt{Bot}$ or
   (b) $\Gamma \vdash \mathtt{R} \Uparrow \mathtt{List(P)}$ with $\Gamma \vdash \mathtt{P} <: \mathtt{Q}$.
3. If $\Gamma \vdash \mathtt{S} \Uparrow \mathtt{Bot}$, then $\Gamma \vdash \mathtt{R} \Uparrow \mathtt{Bot}$.

Proof: Straightforward induction on a derivation of $\Gamma \vdash \mathtt{R} <: \mathtt{S}$. ∎

3.3 Meets and Joins

We write $\Gamma \vdash \mathtt{S} \wedge \mathtt{T} = \mathtt{M}$ for "M is the meet of S and T in context Γ" and $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{J}$ for "J is the join of S and T in Γ." The meet and join operations are defined simultaneously as follows. (Note that some of the cases of the definition overlap. Since it is technically convenient to treat meet and join as functions rather than relations, we stipulate that the first clause that applies must be chosen.)

$$\Gamma \vdash \mathtt{S} \wedge \mathtt{T} \;=\; \begin{cases} \mathtt{S} & \text{if } \Gamma \vdash \mathtt{S} <: \mathtt{T} \\ \mathtt{T} & \text{if } \Gamma \vdash \mathtt{T} <: \mathtt{S} \\ \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{U}})\overline{\mathtt{J}}\rightarrow\mathtt{M} & \text{if } \mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{U}})\overline{\mathtt{V}}\rightarrow\mathtt{P} \\ & \quad \mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{U}})\overline{\mathtt{W}}\rightarrow\mathtt{Q} \\ & \quad \Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{U}} \vdash \overline{\mathtt{V}} \vee \overline{\mathtt{W}} = \overline{\mathtt{J}} \\ & \quad \Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{U}} \vdash \mathtt{P} \wedge \mathtt{Q} = \mathtt{M} \\ \mathtt{List(M)} & \text{if } \mathtt{S} = \mathtt{List(P)} \\ & \quad \mathtt{T} = \mathtt{List(Q)} \\ & \quad \Gamma \vdash \mathtt{P} \wedge \mathtt{Q} = \mathtt{M} \\ \mathtt{Bot} & \text{otherwise} \end{cases}$$

$$\Gamma \vdash \mathtt{S} \vee \mathtt{T} \;=\; \begin{cases} \mathtt{T} & \text{if } \Gamma \vdash \mathtt{S} <: \mathtt{T} \\ \mathtt{S} & \text{if } \Gamma \vdash \mathtt{T} <: \mathtt{S} \\ \mathtt{J} & \text{if } \mathtt{S} = \mathtt{X} \text{ and } \Gamma \vdash \Gamma(\mathtt{X}) \vee \mathtt{T} = \mathtt{J} \\ \mathtt{J} & \text{if } \mathtt{T} = \mathtt{X} \text{ and } \Gamma \vdash \mathtt{S} \vee \Gamma(\mathtt{X}) = \mathtt{J} \\ \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{U}})\overline{\mathtt{M}}\rightarrow\mathtt{J} & \text{if } \mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{U}})\overline{\mathtt{V}}\rightarrow\mathtt{P} \\ & \quad \mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{U}})\overline{\mathtt{W}}\rightarrow\mathtt{Q} \\ & \quad \Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{U}} \vdash \overline{\mathtt{V}} \wedge \overline{\mathtt{W}} = \overline{\mathtt{M}} \\ & \quad \Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{U}} \vdash \mathtt{P} \vee \mathtt{Q} = \mathtt{J} \\ \mathtt{List(J)} & \text{if } \mathtt{S} = \mathtt{List(P)} \\ & \quad \mathtt{T} = \mathtt{List(Q)} \\ & \quad \Gamma \vdash \mathtt{P} \vee \mathtt{Q} = \mathtt{J} \\ \mathtt{Top} & \text{otherwise} \end{cases}$$

Note that $\wedge$ and $\vee$ are total functions: for every Γ, S, and T, there are unique types M and J such that $\Gamma \vdash \mathtt{S} \wedge \mathtt{T} = \mathtt{M}$ and $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{J}$.

Before going any further, let us verify that these definitions do indeed calculate meets and joins in the subtype relation. The argument into two parts: Proposition 3.3.1 shows that the calculated meet is a lower bound of $\mathtt{S}$ and $\mathtt{T}$ and the join is an upper bound; Proposition 3.3.2 then shows that the calculated meet is greater than every common lower bound of $\mathtt{S}$ and $\mathtt{T}$ and the join is less than every common upper bound.

3.3.1 Proposition:

1. If $\Gamma \vdash \mathtt{S} \wedge \mathtt{T} = \mathtt{M}$, then $\Gamma \vdash \mathtt{M} \mathrel{<:} \mathtt{S}$ and $\Gamma \vdash \mathtt{M} \mathrel{<:} \mathtt{T}$.

2. If $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{J}$, then $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{J}$ and $\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{J}$.

Proof: By a straightforward induction on the size of a "derivation" of $\Gamma \vdash \mathtt{S} \wedge \mathtt{T} = \mathtt{M}$ or $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{J}$ (i.e., the number of recursive calls to the definitions of $\wedge$ and $\vee$ needed to calculate $\mathtt{M}$ or $\mathtt{J}$). ∎

3.3.2 Proposition:

1. Suppose that $\Gamma \vdash \mathtt{S} \wedge \mathtt{T} = \mathtt{M}$ and, for some $\mathtt{L}$, that $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{S}$ and $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{T}$. Then $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{M}$.

2. Suppose that $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{J}$ and, for some $\mathtt{U}$, that $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{U}$ and $\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{U}$. Then $\Gamma \vdash \mathtt{J} \mathrel{<:} \mathtt{U}$.

Proof: Simultaneously, by induction on the total size of derivations of $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{S}$ and $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{T}$ (for part 1) or $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{U}$ and $\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{U}$ (part 2).

In both parts, let us deal first with the case where $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$ or $\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{S}$. If $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$, then $\Gamma \vdash \mathtt{S} \wedge \mathtt{T} = \mathtt{S}$ and $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{T}$. But then $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{M}$ and $\Gamma \vdash \mathtt{J} \mathrel{<:} \mathtt{U}$ by assumption. Similarly when $\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{S}$.

To complete the proofs, assume that $\Gamma \vdash \mathtt{S} \not\mathrel{<:} \mathtt{T}$ and $\Gamma \vdash \mathtt{T} \not\mathrel{<:} \mathtt{S}$ and consider the two parts of the proposition in turn.

1. Consider the form of $\mathtt{L}$.

   Case: $\mathtt{L} = \mathtt{Top}$

   Then $\mathtt{S} = \mathtt{Top}$ and $\mathtt{T} = \mathtt{Top}$, so $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$ and this case has already been dealt with.

   Case: $\mathtt{L} = \mathtt{Bot}$

   Immediate.

   Case: $\mathtt{L} = \mathtt{X}$

   There are four subcases to consider, depending on whether rule S-Refl or S-Var was used to derive $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{S}$ and $\Gamma \vdash \mathtt{L} \mathrel{<:} \mathtt{T}$.

   In three of these cases (where either $\mathtt{S}$ or $\mathtt{T}$ or both are exactly $\mathtt{X}$), we have either $\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$ or $\Gamma \vdash \mathtt{T} \mathrel{<:} \mathtt{S}$; these cases have already been dealt with.

   Suppose, on the other hand, that $\Gamma \vdash \Gamma(\mathtt{X}) \mathrel{<:} \mathtt{S}$ and $\Gamma \vdash \Gamma(\mathtt{X}) \mathrel{<:} \mathtt{T}$. Then the induction hypothesis yields $\Gamma \vdash \Gamma(\mathtt{X}) \mathrel{<:} \mathtt{M}$, from which $\Gamma \vdash \mathtt{X} \mathrel{<:} \mathtt{M}$ follows by S-Var.

   Case: $\mathtt{L} = \mathtt{All}(\overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}})\overline{\mathtt{A}}{\rightarrow}\mathtt{B}$

   First, note that, since $\Gamma \vdash \mathtt{S} \not\mathrel{<:} \mathtt{T}$ and $\Gamma \vdash \mathtt{T} \not\mathrel{<:} \mathtt{S}$, neither $\mathtt{S}$ nor $\mathtt{T}$ can be $\mathtt{Top}$, so the only remaining case is where

   $$\begin{array}{l} \mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}})\overline{\mathtt{V}}{\rightarrow}\mathtt{P} \\ \mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}})\overline{\mathtt{W}}{\rightarrow}\mathtt{Q} \end{array}$$

   with

   $$\begin{array}{l} \Gamma, \overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{V}} \mathrel{<:} \overline{\mathtt{A}} \\ \Gamma, \overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}} \vdash \mathtt{B} \mathrel{<:} \mathtt{P} \\ \Gamma, \overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{W}} \mathrel{<:} \overline{\mathtt{A}} \\ \Gamma, \overline{\mathtt{X}}\mathrel{<:}\overline{\mathtt{U}} \vdash \mathtt{B} \mathrel{<:} \mathtt{Q}. \end{array}$$

Also, by the definition of meets, $\mathtt{M} = \mathtt{All}(\overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}})\overline{\mathtt{J}}\rightarrow\mathtt{N}$ with

$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{V}} \vee \overline{\mathtt{W}} = \overline{\mathtt{J}}$$
$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \mathtt{P} \wedge \mathtt{Q} = \mathtt{N}.$$

Applying the induction hypothesis, we obtain

$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{J}} \texttt{ <: } \overline{\mathtt{A}}$$
$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \mathtt{B} \texttt{ <: } \mathtt{N},$$

from which $\Gamma \vdash \mathtt{L} \texttt{ <: } \mathtt{M}$ follows by S-Fun.

Case: $\mathtt{L} = \mathtt{List(A)}$

Similar.

2. First, observe that, by Lemma 3.1.6, there are three possibilities for $\mathtt{S}$—it is either $\mathtt{Bot}$ or a variable whose upper bound is a subtype of $\mathtt{U}$, or else its structure depends on the structure of $\mathtt{U}$—and similarly for $\mathtt{T}$. Let us deal first with the first two possibilities for both $\mathtt{S}$ and $\mathtt{T}$, leaving the structural cases to be considered in detail.

   If $\mathtt{S} = \mathtt{Bot}$ or $\mathtt{T} = \mathtt{Bot}$, then $\Gamma \vdash \mathtt{S} \texttt{ <: } \mathtt{T}$ or $\Gamma \vdash \mathtt{T} \texttt{ <: } \mathtt{S}$; we have dealt with this case already.

   If $\mathtt{S} = \mathtt{Y}$ with $\Gamma \vdash \Gamma(\mathtt{Y}) \texttt{ <: } \mathtt{U}$, then, by the definition of $\vee$, we have $\Gamma \vdash \Gamma(\mathtt{Y}) \vee \mathtt{T} = \mathtt{J}$. The induction hypothesis now yields $\Gamma \vdash \mathtt{J} \texttt{ <: } \mathtt{U}$, and we are finished. Similarly if $\mathtt{T}$ is a variable bounded by $\mathtt{U}$.

   Finally, consider the form of $\mathtt{U}$.

   Case: $\mathtt{U} = \mathtt{Top}$

   Immediate.

   Case: $\mathtt{U} = \mathtt{Bot}$

   Since $\mathtt{S}$ and $\mathtt{T}$ are both subtypes of $\mathtt{Bot}$ and $\mathtt{Bot}$ is a subtype of $\mathtt{S}$ and $\mathtt{T}$, transitivity of subtyping yields $\Gamma \vdash \mathtt{S} \texttt{ <: } \mathtt{T}$: we have dealt with this case already.

   Case: $\mathtt{U} = \mathtt{X}$

   By Lemma 3.1.6 we have $\mathtt{S} = \mathtt{X}$ and $\mathtt{T} = \mathtt{X}$; we have dealt with this case already.

   Case: $\mathtt{U} = \mathtt{All}(\overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}})\overline{\mathtt{A}}\rightarrow\mathtt{B}$

   Then by Lemma 3.1.6 we have

$$\mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}})\overline{\mathtt{V}}\rightarrow\mathtt{P}$$
$$\mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}})\overline{\mathtt{W}}\rightarrow\mathtt{Q}$$

   with

$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{A}} \texttt{ <: } \overline{\mathtt{V}}$$
$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \mathtt{P} \texttt{ <: } \mathtt{B}$$
$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \overline{\mathtt{A}} \texttt{ <: } \overline{\mathtt{W}}$$
$$\Gamma, \overline{\mathtt{X}}\texttt{<:}\overline{\mathtt{U}} \vdash \mathtt{Q} \texttt{ <: } \mathtt{B}.$$

   The rest of the argument proceeds as in the corresponding case in part 1.

Case: $\mathtt{U} = \mathtt{List(A)}$

Similar. ∎

Before we go on to the properties of the typing relation, here are two technical lemmas recording useful facts about how joins interact with subtyping and substitution.

3.3.3 Lemma: If $\Gamma \vdash \mathtt{S} \vee \mathtt{T} = \mathtt{J}$, and if $\Gamma \vdash \mathtt{P} <: \mathtt{S}$ and $\Gamma \vdash \mathtt{Q} <: \mathtt{T}$, then $\Gamma \vdash \mathtt{P} \vee \mathtt{Q} = \mathtt{K}$ with $\Gamma \vdash \mathtt{K} <: \mathtt{J}$.

Proof: Directly from Propositions 3.3.1 and 3.3.2. ∎

3.3.4 Lemma: If $\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{U}}, \Delta \vdash \mathtt{P} \vee \mathtt{Q} = \mathtt{J}$ and $\Gamma \vdash \mathtt{S}_i <: [\mathtt{S}_1/\mathtt{X}_1, \ldots, \mathtt{S}_{i-1}/\mathtt{X}_{i-1}]\mathtt{U}_i$ for each i, then $\Gamma, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P} \vee [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{Q} = \mathtt{K}$ with $\Gamma, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash \mathtt{K} <: [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{J}$.

Proof: From Proposition 3.3.1(2), we know

$$\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{U}}, \Delta \vdash \mathtt{P} <: \mathtt{J}$$
$$\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{U}}, \Delta \vdash \mathtt{Q} <: \mathtt{J}.$$

Lemma 3.1.5 (iterated n times) gives

$$\Gamma, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P} <: [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{J}$$
$$\Gamma, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{Q} <: [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{J}.$$

From these facts, Proposition 3.3.2(2) yields

$$\Gamma, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash \mathtt{K} <: [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{J},$$

as required. ∎

4 More Properties of Subtyping

This section develops some additional properties of subtyping that are not required for the development of the subject reduction property in Section 5. (They play a critical role in the local type argument synthesis algorithm developed in [PT97].)

4.1 Bottom Variables and Rigid Types

As discussed in the introduction, it is important to note that some of the usual properties of presentations of Kernel Fun without `Bot` do not hold for the present system. For instance, $\Gamma \vdash \mathtt{S} <: \mathtt{T}$ and $\Gamma \vdash \mathtt{T} <: \mathtt{S}$ do not imply $\mathtt{S} = \mathtt{T}$ (consider, for example, $\mathtt{X}{<:}\mathtt{Bot} \vdash \mathtt{X} <: \mathtt{Bot}$ and $\mathtt{X}{<:}\mathtt{Bot} \vdash \mathtt{Bot} <: \mathtt{X}$). This fact is inherent in the system with `Bot`, and it substantially complicates the proofs of the properties of the system; this is the cost of the "completeness" that we gain by using `Bot`.

This "loss of rigidity" can be characterized by studying the conditions under which a type will have a singleton equivalence class in the subtype relation. For example, if $\mathtt{X}{<:}\mathtt{Bot}$, then $\mathtt{X}{\rightarrow}\mathtt{X}$ is equivalent, but not identical, to $\mathtt{Bot}{\rightarrow}\mathtt{Bot}$: indeed, its equivalence class has several members. On the other hand, `List(Top)` is only equivalent to itself.

4.1.1 Definition: Formally, let us call a type variable a *bottom variable* if it is bounded by `Bot` or by another bottom variable. Now, let Γ be a context and `S` a type whose free variables are in $dom(\Gamma)$. Say that `S` is *rigid under* Γ if

- $\mathtt{S} = \mathtt{Top}$;
- $\mathtt{S} = \mathtt{Bot}$ and no variable in Γ is bounded by `Bot`;
- $\mathtt{S} = \mathtt{X}$ and `X` is not a bottom variable;
- $\mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}{\rightarrow}\mathtt{T}$ with each $\mathtt{S}_i$ rigid with respect to $\Gamma, \mathtt{X}_1{:}\mathtt{S}_1, \ldots, \mathtt{X}_{i-1}{:}\mathtt{S}_{i-1}$ and `T` rigid under $\Gamma, \overline{\mathtt{X}}{:}\overline{\mathtt{S}}$;

- $\mathtt{S} = \mathtt{List(T)}$ and $\mathtt{T}$ is rigid under Γ.

4.1.2 Lemma: If $\mathtt{S}$ is rigid under Γ, then every type equivalent to $\mathtt{S}$ is syntactically equal to $\mathtt{S}$—i.e., $\Gamma \vdash \mathtt{S} <: \mathtt{T}$ and $\Gamma \vdash \mathtt{T} <: \mathtt{S}$ together imply that $\mathtt{S}$ and $\mathtt{T}$ are identical.

Proof: First, observe (by inspecting the subtyping rules) that:

1. If $\Gamma \vdash \mathtt{X} <: \mathtt{T} <: \mathtt{X}$, then either $\mathtt{T} = \mathtt{X}$ or $\mathtt{X}$ is a bottom variable.

2. If $\Gamma \vdash \mathtt{Bot} <: \mathtt{T} <: \mathtt{Bot}$, then either $\mathtt{T} = \mathtt{Bot}$ or $\mathtt{T}$ is a bottom variable.

The desired result now follows by an easy induction on the total depth of the subtyping derivations $\Gamma \vdash \mathtt{S} <: \mathtt{T}$ and $\Gamma \vdash \mathtt{T} <: \mathtt{S}$, using these facts at the base cases for variables and $\mathtt{Bot}$. ∎

4.2 Variable Elimination

In extensions and applications of Kernel Fun, it is often useful to be able to eliminate all occurrences of a given variable from a type expression by promoting the type expression until we reach a supertype in which this variable does not occur. For example, suppose we wanted to extend the present system with existential types [MP88, CW85]. The usual declarative typing rule for the existential elimination form

$$\frac{\Gamma \vdash \mathtt{e} \in \mathtt{Some(X<:S)T} \qquad \Gamma, \mathtt{X<:S}, \mathtt{x:T} \vdash \mathtt{b} \in \mathtt{B} \qquad \mathtt{X} \notin FV(\mathtt{B})}{\Gamma \vdash \mathtt{open\ e\ as\ [X,x]\ in\ b} \in \mathtt{B}}$$

includes a side condition stipulating that the type $\mathtt{B}$ of the body must be well-formed in a context where $\mathtt{X}$ is not in scope. But the *minimal* type of $\mathtt{b}$ may not have this property; in order to construct a typechecking algorithm for this extension, we need to be able to calculate the smallest supertype of the minimal type that does. (This problem is discussed in more detail in [GP97].)

Another application of variable elimination is the constraint-generation algorithm developed in [PT97]. Given two types $\mathtt{S}$ and $\mathtt{T}$, one of which may include some indeterminate "unification variables," this algorithm calculates the minimal constraints on the unification variables such that $\mathtt{S} <: \mathtt{T}$. When $\mathtt{S}$ and $\mathtt{T}$ are both function types, say $\mathtt{All(\overline{X}<:\overline{P})\overline{Q}{\to}R}$ and $\mathtt{All(\overline{X}<:\overline{A})\overline{B}{\to}C)}$, the algorithm needs to descend through the binding of $\mathtt{X}$, calculate constraints for $\mathtt{\overline{B}<:\overline{P}}$ and $\mathtt{R<:C}$, and then "lift" these constraints back through the binder by eliminating any mention of the variables in $\mathtt{\overline{X}}$.

Formally, we write $\mathtt{S} \Uparrow_\Delta \mathtt{T}$ for the relation "$\mathtt{T}$ is the least supertype of $\mathtt{S}$ in which none of the variables in $dom(\Delta)$ appear free." Fortunately, in the present system, such a type can always be found. For example, suppose $dom(\Delta) = \{\mathtt{X}\}$; then $\mathtt{(X,List(X),Int){\to}X} \Uparrow_\Delta \mathtt{(Bot,List(Bot),Int){\to}Top}$. This property is another crucial reason for choosing the "Kernel Fun" variant of $F_\leq$ rather than the "full $F_\leq$" variant where two polymorphic function types with different upper bounds for their type components are allowed to stand in the subtype relation under appropriate conditions; in the latter system, variables cannot always be eliminated in a most general way [GP97].

The variable-elimination relation can be computed as follows (using the opposite relation $\mathtt{S} \Downarrow_\Delta \mathtt{T}$ recursively):

$$\mathtt{Top} \Uparrow_\Delta \mathtt{Top} \qquad \text{(VU-Top)}$$

$$\mathtt{Bot} \Uparrow_\Delta \mathtt{Bot} \qquad \text{(VU-Bot)}$$

$$\frac{\mathtt{X} \in dom(\Delta) \qquad \Delta(\mathtt{X}) \Uparrow_\Delta \mathtt{T}}{\mathtt{X} \Uparrow_\Delta \mathtt{T}} \qquad \text{(VU-Var-1)}$$

$$\frac{\mathtt{X} \notin dom(\Delta)}{\mathtt{X} \Uparrow_\Delta \mathtt{X}} \qquad \text{(VU-Var-2)}$$

$$\frac{\mathtt{S} \Uparrow_\Delta \mathtt{T}}{\mathtt{List(S)} \Uparrow_\Delta \mathtt{List(T)}} \qquad \text{(VU-List)}$$

$$\frac{FV(\overline{\mathtt{A}}) \cap dom(\Delta) = \emptyset \qquad \overline{\mathtt{S}} \Downarrow_\Delta \overline{\mathtt{S}}' \qquad \mathtt{T} \Uparrow_\Delta \mathtt{T}'}{\mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}\to\mathtt{T} \Uparrow_\Delta \mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}'\to\mathtt{T}'} \quad \text{(VU-FUN-1)}$$

$$\frac{FV(\overline{\mathtt{A}}) \cap dom(\Delta) \neq \emptyset}{\mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}\to\mathtt{T} \Uparrow_\Delta \mathtt{Top}} \quad \text{(VU-FUN-2)}$$

The one subtlety in this definition is the case distinction between VU-Fun-1 and VU-Fun-2. It corresponds to the formulation of the subtyping rule S-Fun, which requires that the bounds of the two quantified types being compared must be *identical*, not just equivalent. We might consider refining S-Fun to allow equivalent bounds, but this would introduce substantial complications at many other points. For example, here, we would have to allow for the possibility of replacing bottom variables with Bot in the VU-Fun rules.

The definition of $\mathtt{S} \Downarrow_\Delta \mathtt{T}$ is exactly dual to $\mathtt{S} \Uparrow_\Delta \mathtt{T}$, except for the first rule for variables:

$$\mathtt{Top} \Downarrow_\Delta \mathtt{Top} \quad \text{(VD-TOP)}$$

$$\mathtt{Bot} \Downarrow_\Delta \mathtt{Bot} \quad \text{(VD-BOT)}$$

$$\frac{\mathtt{X} \in dom(\Delta)}{\mathtt{X} \Downarrow_\Delta \mathtt{Bot}} \quad \text{(VD-VAR-1)}$$

$$\frac{\mathtt{X} \notin dom(\Delta)}{\mathtt{X} \Downarrow_\Delta \mathtt{X}} \quad \text{(VD-VAR-2)}$$

$$\frac{\mathtt{S} \Downarrow_\Delta \mathtt{T}}{\mathtt{List(S)} \Downarrow_\Delta \mathtt{List(T)}} \quad \text{(VD-LIST)}$$

$$\frac{FV(\overline{\mathtt{A}}) \cap dom(\Delta) = \emptyset \qquad \overline{\mathtt{S}} \Uparrow_\Delta \overline{\mathtt{S}}' \qquad \mathtt{T} \Downarrow_\Delta \mathtt{T}'}{\mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}\to\mathtt{T} \Downarrow_\Delta \mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}'\to\mathtt{T}'} \quad \text{(VD-FUN-1)}$$

$$\frac{FV(\overline{\mathtt{A}}) \cap dom(\Delta) \neq \emptyset}{\mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{A}})\overline{\mathtt{S}}\to\mathtt{T} \Downarrow_\Delta \mathtt{Bot}} \quad \text{(VD-FUN-2)}$$

It is easy to check that $\Uparrow_\Delta$ and $\Downarrow_\Delta$ are total functions, for each context Δ. These functions are similar to the ones used in [GP97], but somewhat easier because of the presence of Bot in our type system (this is one of the rare points where the presence of Bot makes things easier!).

4.2.1 Lemma [Soundness of variable elimination]: Suppose S and T are types whose free variables fall in $dom(\Gamma, \Delta, \Sigma)$ and that $FV(\Sigma) \cap dom(\Delta) = \emptyset$.

1. If $\mathtt{S} \Uparrow_\Delta \mathtt{T}$ then $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$ and $\Gamma,\Delta,\Sigma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$.
2. If $\mathtt{S} \Downarrow_\Delta \mathtt{T}$ then $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$ and $\Gamma,\Delta,\Sigma \vdash \mathtt{T} \mathrel{<:} \mathtt{S}$.

Proof: Straightforward induction on variable-elimination derivations. ■

4.2.2 Lemma [Completeness of variable elimination]: Suppose that (Γ,Δ,Σ) is a context such that $FV(\Sigma) \cap dom(\Delta) = \emptyset$.

1. If $\Gamma,\Delta,\Sigma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$ and $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$, then $\mathtt{S} \Uparrow_\Delta \mathtt{R}$ with $\Gamma,\Sigma \vdash \mathtt{R} \mathrel{<:} \mathtt{T}$.
2. If $\Gamma,\Delta,\Sigma \vdash \mathtt{T} \mathrel{<:} \mathtt{S}$ and $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$, then $\mathtt{S} \Downarrow_\Delta \mathtt{R}$ with $\Gamma,\Sigma \vdash \mathtt{T} \mathrel{<:} \mathtt{R}$.

Proof: By induction on the depth of a derivation of $\Gamma,\Delta,\Sigma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$ or $\Gamma,\Delta,\Sigma \vdash \mathtt{T} \mathrel{<:} \mathtt{S}$.

1. Consider the final rule in the given derivation of $\Gamma,\Delta,\Sigma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$.

Case S-TOP: $\mathtt{T} = \mathtt{Top}$

Since $\Uparrow_\Delta$ is a total function, there is some $\mathtt{R}$ such that $\mathtt{S} \Uparrow_\Delta \mathtt{R}$. Now $\Gamma,\Sigma \vdash \mathtt{R} <: \mathtt{Top}$ is immediate by S-TOP.

Case S-BOT: $\mathtt{S} = \mathtt{Bot}$

Then $\mathtt{R} = \mathtt{Bot}$, and $\Gamma,\Sigma \vdash \mathtt{Bot} <: \mathtt{Bot}$ by S-BOT.

Case S-REFL: $\mathtt{S} = \mathtt{T} = \mathtt{X}$

Since $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$, we have $\mathtt{S} \Uparrow_\Delta \mathtt{X}$ by VU-VAR-2 and $\Gamma,\Sigma \vdash \mathtt{S} <: \mathtt{X}$ by S-REFL.

Case S-VAR: $\mathtt{S} = \mathtt{X}$
$\Gamma,\Delta,\Sigma \vdash (\Gamma,\Delta,\Sigma)(\mathtt{X}) <: \mathtt{T}$

There are two cases to consider. If $\mathtt{X} \in dom(\Gamma,\Sigma)$, then $\mathtt{S} \Uparrow_\Delta \mathtt{X}$ by VU-VAR-2 and $\Gamma,\Sigma \vdash \mathtt{X} <: \mathtt{X}$ as above. On the other hand, if $\mathtt{X} \in dom(\Delta)$, then, by VU-VAR-1, $\mathtt{S} \Uparrow_\Delta \mathtt{R}$ and $\Delta(\mathtt{X}) \Uparrow_\Delta \mathtt{R}$. By the induction hypothesis, $\Gamma,\Sigma \vdash \mathtt{R} <: \mathtt{T}$.

Case S-FUN: $\mathtt{S} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{B}})\overline{\mathtt{U}}\rightarrow\mathtt{P}$
$\mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{B}})\overline{\mathtt{V}}\rightarrow\mathtt{Q}$
$\Gamma,\ \Delta,\ (\Sigma,\ \overline{\mathtt{X}}<:\overline{\mathtt{B}}) \vdash \overline{\mathtt{V}} <: \overline{\mathtt{U}}$
$\Gamma,\ \Delta,\ (\Sigma,\ \overline{\mathtt{X}}<:\overline{\mathtt{B}}) \vdash \mathtt{P} <: \mathtt{Q}$

Since $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$, we know in particular that $FV(\overline{\mathtt{B}}) \cap dom(\Delta) = \emptyset$. So rule VU-FUN-1 applies, yielding

$$\overline{\mathtt{U}} \Downarrow_\Delta \overline{\mathtt{U}}'$$
$$\mathtt{P} \Uparrow_\Delta \mathtt{P}'$$
$$\mathtt{R} = \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{B}})\overline{\mathtt{U}}'\rightarrow\mathtt{P}'.$$

By the induction hypothesis, $\Gamma,\ (\Sigma,\ \overline{\mathtt{X}}<:\overline{\mathtt{B}}) \vdash \overline{\mathtt{V}} <: \overline{\mathtt{U}}'$ and $\Gamma,\ (\Sigma,\ \overline{\mathtt{X}}<:\overline{\mathtt{B}}) \vdash \mathtt{P}' <: \mathtt{Q}$. Hence, by S-FUN, $\Gamma,\Sigma \vdash \mathtt{R} <: \mathtt{T}$.

Case S-LIST: $\mathtt{S} = \mathtt{List(P)}$
$\mathtt{T} = \mathtt{List(Q)}$

Straightforward.

2. Consider the final rule in the given derivation of $\Gamma,\Delta,\Sigma \vdash \mathtt{T} <: \mathtt{S}$.

   Case S-TOP: $\mathtt{S} = \mathtt{Top}$

   Then $\mathtt{S} \Downarrow_\Delta \mathtt{Top}$ by VD-TOP and $\Gamma,\Sigma \vdash \mathtt{T} <: \mathtt{Top}$ by S-TOP.

   Case S-BOT: $\mathtt{S} = \mathtt{Bot}$

   Then $\mathtt{S} \Downarrow_\Delta \mathtt{R}$ for some $\mathtt{R}$ (because $\Downarrow_\Delta$ is total), and $\Gamma,\Sigma \vdash \mathtt{Bot} <: \mathtt{R}$ by S-BOT.

   Case S-VAR: $\mathtt{T} = \mathtt{X}$
   $\Gamma,\Delta,\Sigma \vdash (\Gamma,\Delta,\Sigma)(\mathtt{X}) <: \mathtt{S}$

   First, note that $(\Gamma,\Delta,\Sigma)(\mathtt{X}) = (\Gamma,\Sigma)(\mathtt{X})$, since $FV(\mathtt{T}) \cap dom(\Delta) = \emptyset$. Now, by assumption, $FV((\Gamma,\Sigma)(\mathtt{X})) \cap dom(\Delta) = \emptyset$, so the induction hypothesis applies, yielding $\mathtt{S} \Downarrow_\Delta \mathtt{R}$ with $\Gamma,\Sigma \vdash (\Gamma,\Sigma)(\mathtt{X}) <: \mathtt{R}$. By S-VAR, $\Gamma,\Sigma \vdash \mathtt{X} <: \mathtt{R}$, as required.

   Cases S-REFL, S-FUN, and S-LIST:

   Dual to the corresponding cases in part 1. ∎

5 Typing

The typing relation $\Gamma \vdash \mathtt{e} \in \mathtt{T}$ is essentially the standard one, plus some extra rules dealing with Bot. As in the definition of subtyping, we begin with an algorithmic presentation, omitting the usual rule of subsumption ("if $\mathtt{e} \in \mathtt{S}$ and $\mathtt{S} <: \mathtt{T}$, then $\mathtt{e} \in \mathtt{T}$"). The rules below calculate for each typeable term a single *manifest type*, corresponding to its minimal type in the system with subsumption (this correspondence is made precise in Section 5.1). It should be emphasized that this choice is made for purely stylistic reasons (mainly for consistency with the development in [PT97]): it does not affect the set of typeable terms.

In the rules, $\Gamma \vdash \mathtt{e} \Uparrow \mathtt{T}$ abbreviates "$\Gamma \vdash \mathtt{e} \in \mathtt{S}$" and $\Gamma \vdash \mathtt{S} \Uparrow \mathtt{T}$; similarly, $\Gamma \vdash \mathtt{e} \in \mathtt{S} <: \mathtt{T}$ abbreviates "$\Gamma \vdash \mathtt{e} \in \mathtt{S}$ and $\Gamma \vdash \mathtt{S} <: \mathtt{T}$."

The typing rules for variables and constants are standard.

$$\Gamma \vdash \mathtt{x} \in \Gamma(\mathtt{x}) \tag{T-Var}$$

$$\Gamma \vdash \mathtt{c} \in \textit{type-of-const}(\mathtt{c}) \tag{T-Con}$$

The rule for (multi-)abstractions combines the usual rules for term and type abstractions.

$$\frac{\Gamma, \overline{\mathtt{X}}<:\overline{\mathtt{B}}, \overline{\mathtt{x}}:\overline{\mathtt{S}} \vdash \mathtt{e} \in \mathtt{T}}{\Gamma \vdash \mathtt{fun}[\overline{\mathtt{X}}<:\overline{\mathtt{B}}]\,(\overline{\mathtt{x}}:\overline{\mathtt{S}})\mathtt{e} \in \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{B}})\overline{\mathtt{S}}\to\mathtt{T}} \tag{T-Abs}$$

Similarly, the rule for (multi-)applications combines the usual application and type application rules of Kernel Fun. We calculate the type of the function e and promote it (if necessary), to an explicit function type $\mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{B}})\overline{\mathtt{S}}\to\mathtt{R}$. We then check that the provided type arguments are subtypes of the upper bounds $\overline{\mathtt{B}}$ and that the provided term arguments have the expected types. If all these constraints are satisfied, then the result type is found by substituting the actual type arguments into R.

$$\frac{\Gamma \vdash \mathtt{e} \Uparrow \mathtt{All}(\overline{\mathtt{X}}<:\overline{\mathtt{B}})\overline{\mathtt{S}}\to\mathtt{R} \qquad \Gamma \vdash \mathtt{T}_i <: [\mathtt{T}_1/\mathtt{X}_1 \ldots \mathtt{T}_{i-1}/\mathtt{X}_{i-1}]\mathtt{B}_i \qquad \Gamma \vdash \overline{\mathtt{a}} \in \overline{\mathtt{U}} <: [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{S}}}{\Gamma \vdash \mathtt{e}[\overline{\mathtt{T}}]\,(\overline{\mathtt{a}}) \in [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{R}} \tag{T-App}$$

The typing rule for case checks that the first argument is actually a list, calculates types for the nil and cons branches, and yields the least common supertype of these as final result type.

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{e}_1 \Uparrow \mathtt{List(S)} \qquad \Gamma \vdash \mathtt{e}_2 \in \mathtt{E}_2 \\ \Gamma, \mathtt{x}:\mathtt{S}, \mathtt{y}:\mathtt{List(S)} \vdash \mathtt{e}_3 \in \mathtt{E}_3 \qquad \Gamma \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{J}\end{array}}{\Gamma \vdash \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}\to\mathtt{e}_2 \mid \mathtt{cons(x,y)}\to\mathtt{e}_3 \in \mathtt{J}} \tag{T-Case}$$

To finish the definition of the typing relation, two more rules are needed. To see why, note that, unlike ordinary Kernel Fun without Bot, we can have, for example, $\Gamma \vdash \mathtt{S} <: \mathtt{List(U)}$ even though the least non-variable supertype of S does not have the form List(V): the least non-variable supertype of S can also be Bot, which can be promoted to *any* list type.

$$\frac{\Gamma \vdash \mathtt{e} \Uparrow \mathtt{Bot} \qquad \Gamma \vdash \overline{\mathtt{a}} \in \overline{\mathtt{U}}}{\Gamma \vdash \mathtt{e}(\overline{\mathtt{T}},\overline{\mathtt{a}}) \in \mathtt{Bot}} \tag{T-App-Bot}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{e}_1 \Uparrow \mathtt{Bot} \qquad \Gamma \vdash \mathtt{e}_2 \in \mathtt{E}_2 \\ \Gamma, \mathtt{x}:\mathtt{Bot}, \mathtt{y}:\mathtt{List(Bot)} \vdash \mathtt{e}_3 \in \mathtt{E}_3 \qquad \Gamma \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{J}\end{array}}{\Gamma \vdash \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}\to\mathtt{e}_2 \mid \mathtt{cons(x,y)}\to\mathtt{e}_3 \in \mathtt{J}} \tag{T-Case-Bot}$$

5.1 Theorem [Uniqueness of manifest types]: Each typeable term has a unique type: if $\Gamma \vdash \mathtt{e} \in \mathtt{S}$ and $\Gamma \vdash \mathtt{e} \in \mathtt{T}$, then $\mathtt{S} = \mathtt{T}$.

Proof: By the syntax-directedness of the typing rules and the fact that $\Uparrow$ and $\vee$ are total functions. ∎

5.1 Subsumption

The technicalities of the proof of subject reduction in Section 5.3 turn out to be simpler if we work in an extended system formed by adjoining to the above rules the familiar rule of *subsumption*:

$$\frac{\Gamma \vdash^{\leq:} \mathtt{e} \in \mathtt{T} \qquad \Gamma \vdash \mathtt{S} <: \mathtt{T}}{\Gamma \vdash^{\leq:} \mathtt{e} \in \mathtt{T}} \tag{T-Sub}$$

We write $\Gamma \vdash^{\leq:} \mathtt{e} \in \mathtt{T}$ for derivations in the system with subsumption.

The remainder of this section is devoted to showing that, as usual, this addition does not change the basic character of the system: the system without subsumption can be regarded as an algorithm for finding minimal typings in the system with subsumption.

5.1.1 Definition: The *weight* of a derivation $\Gamma \vdash^{\leq:} \mathtt{e} \in \mathtt{T}$ is $\omega \cdot n + s$, where s is the number of instances of T-Sub used in the derivation and n is the number of instances of all the other rules.

5.1.2 Lemma [Narrowing for term variables]: If $\Gamma, \mathtt{x}{:}\mathtt{B}, \Delta \vdash^{\leq:} \mathtt{e} \in \mathtt{T}$ and $\Gamma \vdash \mathtt{A} <: \mathtt{B}$, then $\Gamma, \mathtt{x}{:}\mathtt{A}, \Delta \vdash^{\leq:} \mathtt{e} \in \mathtt{T}$. Moreover, if the weight of the original derivation is $\omega \cdot n + s$, then the weight of the new derivation is $\omega \cdot n + s'$ for some s'. (That is, the new derivation differs from the old one only in possibly having some extra uses of T-Sub.)

Proof: Straightforward induction, using T-Sub as necessary at the leaves. ∎

5.1.3 Theorem [Admissibility of subsumption]: If $\Gamma \vdash^{\leq:} \mathtt{e} \in \mathtt{T}$, then $\Gamma \vdash \mathtt{e} \in \mathtt{S}$ for some $\mathtt{S}$ with $\Gamma \vdash \mathtt{S} <: \mathtt{T}$.

Proof: By induction on the weight of a derivation of $\Gamma \vdash^{\leq:} \mathtt{e} \in \mathtt{T}$, with a case analysis on the last rule used in the derivation.

Case T-Var, T-Con:

Immediate.

Case T-Abs:

Straightforward.

Case T-App: $\mathtt{e} = \mathtt{e}_1(\overline{\mathtt{T}},\overline{\mathtt{a}})$
$\Gamma \vdash^{\leq:} \mathtt{e}_1 \in \mathtt{E}_1$
$\Gamma \vdash \mathtt{E}_1 \Uparrow \mathtt{All}(\overline{\mathtt{X}}{<:}\overline{\mathtt{B}})\overline{\mathtt{S}}{\rightarrow}\mathtt{R}$
$\Gamma \vdash \mathtt{T}_i <: [\mathtt{T}_1/\mathtt{X}_1 \ldots \mathtt{T}_{i-1}/\mathtt{X}_{i-1}]\mathtt{B}_i$ for each i
$\Gamma \vdash^{\leq:} \overline{\mathtt{a}} \in \overline{\mathtt{U}}$
$\Gamma \vdash \overline{\mathtt{U}} <: [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{S}}$
$\mathtt{T} = [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{R}}$

By the induction hypothesis,

$\Gamma \vdash \mathtt{e}_1 \in \mathtt{D} <: \mathtt{E}_1$
$\Gamma \vdash \overline{\mathtt{a}} \in \overline{\mathtt{A}} <: \overline{\mathtt{U}}$.

By Lemma 3.2.2(1), there are two cases to consider.

1. $\Gamma \vdash \mathtt{D} \Uparrow \mathtt{Bot}$. Then we have $\Gamma \vdash \mathtt{e}_1 \Uparrow \mathtt{Bot}$ and, by T-App-Bot, $\Gamma \vdash \mathtt{e}_1(\overline{\mathtt{T}},\overline{\mathtt{a}}) \in \mathtt{Bot}$. By S-Bot, we have $\Gamma \vdash \mathtt{Bot} <: \mathtt{T}$, as required.

2. $\Gamma \vdash \mathtt{D} \Uparrow \mathtt{All}(\overline{\mathtt{X}}{<:}\overline{\mathtt{B}})\overline{\mathtt{P}}{\rightarrow}\mathtt{Q}$, with

   $\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{B}} \vdash \overline{\mathtt{S}} <: \overline{\mathtt{P}}$
   $\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{B}} \vdash \mathtt{Q} <: \mathtt{R}$.

Iterating Lemma 3.1.5 n times, we obtain

$$\begin{array}{ll} \Gamma, \mathtt{X}_1\texttt{<:}\mathtt{B}_1,\ \ldots,\ \mathtt{X}_n\texttt{<:}\mathtt{B}_n \vdash \overline{\mathtt{S}} \texttt{ <: } \overline{\mathtt{P}} & \text{by assumption} \\ \Gamma, \mathtt{X}_2\texttt{<:}\mathtt{B}_2,\ \ldots,\ \mathtt{X}_n\texttt{<:}\mathtt{B}_n \vdash [\mathtt{T}_1/\mathtt{X}_1]\overline{\mathtt{S}} \texttt{ <: } [\mathtt{T}_1/\mathtt{X}_1]\overline{\mathtt{P}} & \text{by 3.1.5} \\ \vdots & \\ \Gamma \vdash [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{S}} \texttt{ <: } [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{P}} & \text{by 3.1.5,} \end{array}$$

from which transitivity yields

$$\Gamma \vdash \mathtt{A} \texttt{ <: } [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{P}}.$$

Also, by the transitivity of subtyping, we have

$$\Gamma \vdash \overline{\mathtt{A}} \texttt{ <: } [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\overline{\mathtt{S}},$$

and, by T-App,

$$\Gamma \vdash \mathtt{e}_1\,[\overline{\mathtt{T}}]\,(\overline{\mathtt{a}}) \in [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{Q}.$$

Finally, iterating Lemma 3.1.5 as above, we obtain $\Gamma \vdash [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{Q} \texttt{ <: } [\overline{\mathtt{T}}/\overline{\mathtt{X}}]\mathtt{R}$, which completes this case of the argument.

Case T-Case: $\mathtt{e} = \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}{\rightarrow}\mathtt{e}_2\ \texttt{|}\ \mathtt{cons(x,y)}{\rightarrow}\mathtt{e}_3$
$\Gamma \vdash^{\leq:} \mathtt{e}_1 \in \mathtt{E}_1 \Uparrow \mathtt{List(S)}$
$\Gamma \vdash^{\leq:} \mathtt{e}_2 \in \mathtt{E}_2$
$\Gamma, \mathtt{x}{:}\mathtt{S}, \mathtt{y}{:}\mathtt{List(S)} \vdash^{\leq:} \mathtt{e}_3 \in \mathtt{E}_3$
$\Gamma \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{T}$

By the induction hypothesis,

$$\begin{array}{l} \Gamma \vdash \mathtt{e}_1 \in \mathtt{D}_1 \texttt{ <: } \mathtt{E}_1 \\ \Gamma \vdash \mathtt{e}_2 \in \mathtt{D}_2 \texttt{ <: } \mathtt{E}_2. \end{array}$$

By Lemma 3.2.2(2), there are two cases to consider:

1. $\Gamma \vdash \mathtt{D}_1 \Uparrow \mathtt{Bot}$. Then, by Lemma 5.1.2 (twice),

$$\Gamma, \mathtt{x}{:}\mathtt{Bot}, \mathtt{y}{:}\mathtt{List(Bot)}\Gamma \vdash^{\leq:} \mathtt{e}_3 \in \mathtt{E}_3,$$

   by a derivation differing from the given derivation of $\Gamma, \mathtt{x}{:}\mathtt{S}, \mathtt{y}{:}\mathtt{List(S)}\Gamma \vdash^{\leq:} \mathtt{e}_3 \in \mathtt{E}_3$ in at most the addition of some instances of T-Sub and therefore of weight smaller than the weight of the whole original derivation. The induction hypothesis thus applies, giving

$$\Gamma, \mathtt{x}{:}\mathtt{Bot}, \mathtt{y}{:}\mathtt{List(Bot)} \vdash \mathtt{e}_3 \in \mathtt{D}_3 \texttt{ <: } \mathtt{E}_3.$$

   By Lemma 3.3.3, $\Gamma \vdash \mathtt{D}_2 \vee \mathtt{D}_3 = \mathtt{K}$ with $\Gamma \vdash \mathtt{K} \texttt{ <: } \mathtt{T}$. Finally, by T-Case-Bot,

$$\Gamma \vdash \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}{\rightarrow}\mathtt{e}_2\ \texttt{|}\ \mathtt{cons(x,y)}{\rightarrow}\mathtt{e}_3 \in \mathtt{K}.$$

2. $\Gamma \vdash \mathtt{D}_1 \Uparrow \mathtt{List(P)}$ with $\Gamma \vdash \mathtt{P} \texttt{ <: } \mathtt{S}$. Using Lemma 5.1.2 as above, we obtain

$$\Gamma, \mathtt{x}{:}\mathtt{P}, \mathtt{y}{:}\mathtt{List(P)} \vdash^{\leq:} \mathtt{e}_3 \in \mathtt{E}_3,$$

   and, by the induction hypothesis,

$$\Gamma, \mathtt{x}{:}\mathtt{P}, \mathtt{y}{:}\mathtt{List(P)} \vdash \mathtt{e}_3 \in \mathtt{D}_3 \texttt{ <: } \mathtt{E}_3.$$

   By Lemma 3.3.3, $\Gamma \vdash \mathtt{D}_2 \vee \mathtt{D}_3 = \mathtt{K}$ with $\Gamma \vdash \mathtt{K} \texttt{ <: } \mathtt{T}$. Finally, by T-Case,

$$\Gamma \vdash \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}{\rightarrow}\mathtt{e}_2\ \texttt{|}\ \mathtt{cons(x,y)}{\rightarrow}\mathtt{e}_3 \in \mathtt{K}.$$

Case T-APP-BOT: $\mathtt{e} = \mathtt{e}_1(\overline{\mathtt{T}},\overline{\mathtt{a}})$
$\Gamma \vdash^{<:} \mathtt{e}_1 \in \mathtt{E}_1 \Uparrow \mathtt{Bot}$
$\Gamma \vdash^{<:} \overline{\mathtt{a}} \in \overline{\mathtt{U}}$

Similar to the following case.

Case T-CASE-BOT: $\mathtt{e} = \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}{\to}\mathtt{e}_2 \mid \mathtt{cons(x,y)}{\to}\mathtt{e}_3$
$\Gamma \vdash^{<:} \mathtt{e}_1 \Uparrow \mathtt{E}_1$
$\Gamma \vdash^{<:} \mathtt{e}_2 \in \mathtt{E}_2$
$\Gamma, \mathtt{x{:}Bot}, \mathtt{y{:}List(Bot)} \vdash^{<:} \mathtt{e}_3 \in \mathtt{E}_3$
$\Gamma \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{J}$

By the induction hypothesis,

$$\Gamma \vdash \mathtt{e}_1 \in \mathtt{D}_1 \mathrel{<:} \mathtt{Bot}$$
$$\Gamma \vdash \mathtt{e}_2 \in \mathtt{D}_2 \mathrel{<:} \mathtt{E}_2$$
$$\Gamma, \mathtt{x{:}Bot}, \mathtt{y{:}List(Bot)} \vdash \mathtt{e}_3 \in \mathtt{D}_3 \mathrel{<:} \mathtt{E}_3.$$

By Lemma 3.2.2(3), $\Gamma \vdash \mathtt{D}_1 \Uparrow \mathtt{Bot}$. By Lemma 3.3.3, $\Gamma \vdash \mathtt{D}_2 \vee \mathtt{D}_3 = \mathtt{K}$ with $\Gamma \vdash \mathtt{K} \mathrel{<:} \mathtt{T}$. The result follows by T-CASE-BOT.

Case T-SUB: $\Gamma \vdash^{<:} \mathtt{e} \in \mathtt{S}$
$\Gamma \vdash \mathtt{S} \mathrel{<:} \mathtt{T}$

Direct from the induction hypothesis. ∎

5.2 Reduction

For the proof of subject reduction, we choose the usual notion of typed β-reduction. Formally, the one-step reduction relation $\longrightarrow^1_\beta$ is the least relation closed under the following computation rules

$$\mathtt{(fun[\overline{X}{<:}\overline{U}](\overline{x}{:}\overline{T})b)[\overline{S}](\overline{a})} \quad \longrightarrow^1_\beta \quad [\overline{\mathtt{S}}/\overline{\mathtt{X}}, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{b} \qquad \text{(R-BETA)}$$

$$\mathtt{case\ nil(V)\ of\ nil}{\to}\mathtt{e}_2 \mid \mathtt{cons(x,y)}{\to}\mathtt{e}_3 \quad \longrightarrow^1_\beta \quad \mathtt{e}_2 \qquad \text{(R-CASE-NIL)}$$

$$\mathtt{case\ cons(V,a,b)\ of\ nil}{\to}\mathtt{e}_2 \mid \mathtt{cons(x,y)}{\to}\mathtt{e}_3 \quad \longrightarrow^1_\beta \quad [\mathtt{a/x}, \mathtt{b/y}]\mathtt{e}_3 \qquad \text{(R-CASE-CONS)}$$

plus congruence rules for all of the term constructors.

5.3 Subject Reduction

We prove subject reduction for the system with the rule of subsumption, from which subject reduction in the original (algorithmic) system will follow as a corollary. As usual, most of the hard work lies in establishing a substitution lemma for the typing relation (Lemma 5.3.1). Using this, the actual argument for subject reduction (Theorem 5.3.2) is a fairly straightforward induction.

5.3.1 Lemma [Substitution]: If

$$\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}{:}\overline{\mathtt{V}}, \Delta \vdash^{<:} \mathtt{e} \in \mathtt{E}$$

and

$$\Gamma \vdash \mathtt{S}_i \mathrel{<:} [\mathtt{S}_1/\mathtt{X}_1 \ldots \mathtt{S}_{i-1}/\mathtt{X}_{i-1}]\mathtt{U}_i$$
$$\Gamma \vdash^{<:} \overline{\mathtt{a}} \in \overline{\mathtt{A}} \mathrel{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{V}},$$

then

$$\Gamma, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}}, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{e} \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}.$$

Proof: By induction on the weight of a derivation of $\Gamma, \overline{\mathtt{X}}{<:}\overline{\mathtt{B}}, \overline{\mathtt{x}}{:}\overline{\mathtt{V}}, \Delta \vdash^{<:} \mathtt{e} \in \mathtt{E}$.

Case T-VAR: $\mathtt{e} = \mathtt{x}$
$\mathtt{E} = (\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta)(\mathtt{x})$

Straightforward.

Case T-CON: $\mathtt{e} = \mathtt{c}$
$\mathtt{E} = \textit{type-of-const}(\mathtt{c})$

Immediate.

Case T-ABS:

Straightforward.

Case T-APP: $\mathtt{e} = \mathtt{f}(\overline{\mathtt{T}},\overline{\mathtt{b}})$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash^{<:} \mathtt{f} \in \mathtt{F}$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash \mathtt{F} \Uparrow \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{P}})\overline{\mathtt{Q}}\to\mathtt{R}$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash \mathtt{T}_i \mathrel{\mathtt{<:}} [\mathtt{T}_1/\mathtt{Y}_1 \ldots \mathtt{T}_{i-1}/\mathtt{Y}_{i-1}]\mathtt{P}_i$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash^{<:} \overline{\mathtt{b}} \in \overline{\mathtt{B}}$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash \overline{\mathtt{B}} \mathrel{\mathtt{<:}} [\overline{\mathtt{T}}/\overline{\mathtt{Y}}]\overline{\mathtt{Q}}$
$\mathtt{E} = [\overline{\mathtt{T}}/\overline{\mathtt{Y}}]\mathtt{R}$

Adding an instance of T-SUB to the first subderivation, we obtain $\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash^{<:} \mathtt{f} \in \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}\overline{\mathtt{P}})\overline{\mathtt{Q}}\to\mathtt{R}$ by a derivation of smaller weight than the given derivation of $\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash \mathtt{f}[\overline{\mathtt{T}}](\overline{\mathtt{b}}) \in \mathtt{E}$. The induction hypothesis applies to this derivation, yielding

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{f} \in \mathtt{All}(\overline{\mathtt{Y}}\mathtt{<:}[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{P}})[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{Q}}\to[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{R}.$$

Similarly,

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\overline{\mathtt{b}} \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]([\overline{\mathtt{T}}/\overline{\mathtt{Y}}]\overline{\mathtt{Q}}),$$

i.e.,

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\overline{\mathtt{b}} \in [([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{T}})/\overline{\mathtt{Y}}]([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{Q}}).$$

Furthermore, iterating Lemma 3.1.5 n times (beginning from the hypothesis $\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash \mathtt{T}_i \mathrel{\mathtt{<:}} [\mathtt{T}_1/\mathtt{Y}_1 \ldots \mathtt{T}_{i-1}/\mathtt{Y}_{i-1}]\mathtt{P}_i$) yields

$$\Gamma,\, \overline{\mathtt{x}}\mathtt{:}[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{V},\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{T}_i \mathrel{\mathtt{<:}} [([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{T}_1)/\mathtt{Y}_1 \ldots ([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{T}_{i-1})/\mathtt{Y}_{i-1}]([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P}_i),$$

from which strengthening for term variables (Lemma 3.1.2) gives

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{T}_i \mathrel{\mathtt{<:}} [([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{T}_1)/\mathtt{Y}_1 \ldots ([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{T}_{i-1})/\mathtt{Y}_{i-1}]([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P}_i).$$

Rule T-APP now yields

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} ([\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{f})([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{T}},\, [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\overline{\mathtt{b}}) \in [([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{T}})/\overline{\mathtt{Y}}]([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{R}),$$

i.e.,

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{e} \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}.$$

Case T-CASE: $\mathtt{e} = \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}\to\mathtt{e}_2\ \mid\ \mathtt{cons(x,y)}\to\mathtt{e}_3$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash^{<:} \mathtt{e}_1 \in \mathtt{E}_1 \Uparrow \mathtt{List(P)}$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash^{<:} \mathtt{e}_2 \in \mathtt{E}_2$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta,\, \mathtt{x}\mathtt{:}\mathtt{P},\, \mathtt{y}\mathtt{:}\mathtt{List(P)} \vdash^{<:} \mathtt{e}_3 \in \mathtt{E}_3$
$\Gamma,\, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}},\, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}},\, \Delta \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{E}$

Using T-SUB and the induction hypothesis as before, we obtain

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{e}_1 \in \mathtt{List}([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P})$$
$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{e}_2 \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}_2$$
$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta,\, \mathtt{x}\mathtt{:}[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P},\, \mathtt{y}\mathtt{:}\mathtt{List}([\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{P}) \vdash^{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}},\, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{e}_3 \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}_3.$$

By Lemma 3.3.4,

$$\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}_2 \vee [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}_3 = \mathtt{J}$$

with $\Gamma,\, [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash \mathtt{J} \mathrel{\mathtt{<:}} [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}$, from which the result follows by T-SUB.

Case T-App-Bot: $\mathtt{e} = \mathtt{f}(\overline{\mathtt{T}},\overline{\mathtt{b}})$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash^{\leq} \mathtt{f} \Uparrow \mathtt{Bot}$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash^{\leq} \overline{\mathtt{a}} \in \overline{\mathtt{P}}$

Straightforward.

Case T-Case-Bot: $\mathtt{e} = \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}\rightarrow\mathtt{e}_2\ \mathtt{|}\ \mathtt{cons(x,y)}\rightarrow\mathtt{e}_3$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash^{\leq} \mathtt{e}_1 \Uparrow \mathtt{Bot}$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash^{\leq} \mathtt{e}_2 \in \mathtt{E}_2$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta, \mathtt{x:Bot}, \mathtt{y:List(Bot)} \vdash^{\leq} \mathtt{e}_3 \in \mathtt{E}_3$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{E}$

Straightforward.

Case T-Sub: $\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash^{\leq} \mathtt{e} \in \mathtt{U}$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}}, \Delta \vdash \mathtt{U} \mathtt{<:} \mathtt{E}$

By the induction hypothesis, Γ, $[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash^{\leq} [\overline{\mathtt{S}}/\overline{\mathtt{X}}, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{e} \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{U}$. Furthermore, iterating Lemma 3.1.5 n times yields Γ, $\overline{\mathtt{x}}\mathtt{:}[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{V}}$, $[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{U} \mathtt{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}$, from which Lemma 3.1.2 gives Γ, $[\overline{\mathtt{S}}/\overline{\mathtt{X}}]\Delta \vdash [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{U} \mathtt{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{E}$. The result follows by T-Sub. ∎

5.3.2 Theorem [Subject reduction for the system with subsumption]: If $\Gamma \vdash^{\leq} \mathtt{e} \in \mathtt{T}$ and $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$, then $\Gamma \vdash^{\leq} \mathtt{e}' \in \mathtt{T}$.

Proof: By induction on a derivation of $\Gamma \vdash^{\leq} \mathtt{e} \in \mathtt{T}$.

Case T-Var, T-Con: $\mathtt{e} = \mathtt{x}$ or $\mathtt{e} = \mathtt{c}$

Can't happen, since there is no $\mathtt{e}'$ such that $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$.

Case T-Abs: $\mathtt{e} = \mathtt{fun}[\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}]\,(\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}})\mathtt{b}$
$\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}} \vdash^{\leq} \mathtt{b} \in \mathtt{B}$
$\mathtt{T} = \mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}})\overline{\mathtt{V}}\rightarrow\mathtt{B}$

By the definition of $\longrightarrow^1_\beta$, it must be the case that $\mathtt{e}' = \mathtt{fun}[\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}]\,(\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}})\mathtt{b}'$ for some $\mathtt{b}'$ with $\mathtt{b} \longrightarrow^1_\beta \mathtt{b}'$. The result then follows by induction.

Case T-App: $\mathtt{e} = \mathtt{f}\,[\overline{\mathtt{S}}]\,(\overline{\mathtt{a}})$
$\Gamma \vdash^{\leq} \mathtt{f} \Uparrow \mathtt{All}(\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}})\overline{\mathtt{V}}\rightarrow\mathtt{B}$
$\Gamma \vdash \mathtt{S}_i \mathtt{<:} [\mathtt{S}_1/\mathtt{X}_1 \ldots \mathtt{S}_{i-1}/\mathtt{X}_{i-1}]\mathtt{U}_i$
$\Gamma \vdash^{\leq} \overline{\mathtt{a}} \in \overline{\mathtt{A}} \mathtt{<:} [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\overline{\mathtt{V}}$
$\mathtt{T} = [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{B}$

There are two subcases to consider:

1. If $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$ by some congruence rule, then either $\mathtt{f} \longrightarrow^1_\beta \mathtt{f}'$ or $\mathtt{a}_i \longrightarrow^1_\beta \mathtt{a}_i{}'$ for some i and the argument proceeds by induction.

   (It is here that we see the benefit of working in the system with subsumption. If we were proving subject reduction directly for the system without subsumption, then the induction hypothesis would tell us that the subexpression in which the reduction occurred might decrease, leading to a somewhat tricky case analysis at this point.)

2. If $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$ by R-Beta, then

   $$\mathtt{f} = \mathtt{fun}[\overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}]\,(\overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}})\mathtt{b}$$
   $$\mathtt{e}' = [\overline{\mathtt{S}}/\overline{\mathtt{X}}, \overline{\mathtt{a}}/\overline{\mathtt{x}}]\mathtt{b}.$$

   From T-Abs (using the fact that the typing relation is syntax-directed), we have $\Gamma, \overline{\mathtt{X}}\mathtt{<:}\overline{\mathtt{U}}, \overline{\mathtt{x}}\mathtt{:}\overline{\mathtt{V}} \vdash \mathtt{b} \in \mathtt{B}$. Lemma 5.3.1 now yields $\Gamma \vdash^{\leq} \mathtt{e}' \in [\overline{\mathtt{S}}/\overline{\mathtt{X}}]\mathtt{B}$, as required.

Case T-CASE: $\mathtt{e} = \mathtt{case}\ \mathtt{e}_1\ \mathtt{of}\ \mathtt{nil}\!\rightarrow\!\mathtt{e}_2\ \mid\ \mathtt{cons(x,y)}\!\rightarrow\!\mathtt{e}_3$
$\Gamma \overset{\leq:}{\vdash} \mathtt{e}_1 \Uparrow \mathtt{List(S)}$
$\Gamma \overset{\leq:}{\vdash} \mathtt{e}_2 \in \mathtt{E}_2$
$\Gamma, \mathtt{x}\!:\!\mathtt{S}, \mathtt{y}\!:\!\mathtt{List(S)} \overset{\leq:}{\vdash} \mathtt{e}_3 \in \mathtt{E}_3$
$\Gamma \vdash \mathtt{E}_2 \vee \mathtt{E}_3 = \mathtt{T}$

This time there are three subcases to consider:

1. If $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$ by some congruence rule, the argument proceeds by induction.

2. If $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$ by R-CASE-NIL, then $\mathtt{e}_1 = \mathtt{nil(W)}$ and $\mathtt{e}' = \mathtt{e}_2$. By the soundness of the definition of joins (Proposition 3.3.1), $\Gamma \vdash \mathtt{E}_2 <: \mathtt{T}$, from which T-SUB yields $\Gamma \overset{\leq:}{\vdash} \mathtt{e}_2 \in \mathtt{T}$, as required.

3. If $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$ by R-CASE-CONS, then $\mathtt{e}_1 = \mathtt{cons(W,a,b)}$ and $\mathtt{e}' = [\mathtt{a}/\mathtt{x}, \mathtt{b}/\mathtt{y}]\mathtt{e}_3$. From T-CON and T-APP,

   $\Gamma \overset{\leq:}{\vdash} \mathtt{a} \in \mathtt{A} <: \mathtt{W}$
   $\Gamma \overset{\leq:}{\vdash} \mathtt{b} \in \mathtt{B} <: \mathtt{List(W)}$
   $\mathtt{W} = \mathtt{S}$,

   from which Lemma 5.3.1 gives $\Gamma \overset{\leq:}{\vdash} [\mathtt{a}/\mathtt{x}, \mathtt{b}/\mathtt{y}]\mathtt{e}_3 \in \mathtt{E}_3$. By the soundness of the definition of joins, $\Gamma \vdash \mathtt{E}_3 <: \mathtt{T}$, and the result again follows by T-SUB.

Case T-APP-BOT: $\mathtt{e} = \mathtt{f}(\overline{\mathtt{S}},\overline{\mathtt{a}})$
$\Gamma \overset{\leq:}{\vdash} \mathtt{f} \in \mathtt{F} \Uparrow \mathtt{Bot}$

By Lemma 3.2.2(3), $\mathtt{F} = \mathtt{Bot}$. Inspection of the conclusions of the typing rules now reveals that $\mathtt{f}$ cannot have the form $\mathtt{fun}[\overline{\mathtt{X}}<:\overline{\mathtt{U}}](\overline{\mathtt{x}}:\overline{\mathtt{V}})\mathtt{b}$ (no well-typed abstraction has type $\mathtt{Bot}$), so $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$ by some congruence rule, not R-BETA. The argument proceeds by induction.

Case T-CASE-BOT:

Similar.

Case T-SUB: $\Gamma \overset{\leq:}{\vdash} \mathtt{e} \in \mathtt{S}$
$\Gamma \vdash \mathtt{S} <: \mathtt{T}$

By the induction hypothesis and T-SUB. ∎

5.3.3 Corollary [Subject reduction]: If $\Gamma \vdash \mathtt{e} \in \mathtt{T}$ and $\mathtt{e} \longrightarrow^1_\beta \mathtt{e}'$, then $\Gamma \vdash \mathtt{e}' \in \mathtt{S}$ for some $\mathtt{S}$ with $\Gamma \vdash \mathtt{S} <: \mathtt{T}$.

Acknowledgements

The system studied here arose during work on type inference for languages combining impredicative polymorphism and subtyping with David Turner, who also joined in a number of discussions of the properties developed here. This work was partially supported by NSF grant CCR-9701826, *Principled Foundations for Programming with Objects.*

References

[AC93] Roberto M. Amadio and Luca Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993. A preliminary version appeared in POPL '91 (pp. 104–118) and as DEC Systems Research Center Research Report number 62, August 1990.

[ACV96] Martín Abadi, Luca Cardelli, and Ramesh Viswanathan. An interpretation of objects and object types. In *Principles of Programming Languages*, pages 396–409, 1996.

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

[CG92] Pierre-Louis Curien and Giorgio Ghelli. Coherence of subsumption: Minimum typing and type-checking in $F_{\leq}$. *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, 1994).

[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.

[CM91] Luca Cardelli and John Mitchell. Operations on records. *Mathematical Structures in Computer Science*, 1:3–48, 1991. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press, 1994); available as DEC Systems Research Center Research Report #48, August, 1989, and in the proceedings of MFPS '89, Springer LNCS volume 442.

[CMMS94] Luca Cardelli, Simone Martini, John C. Mitchell, and Andre Scedrov. An extension of system F with subtyping. *Information and Computation*, 109(1–2):4–56, 1994. A preliminary version appeared in TACS '91 (Sendai, Japan, pp. 750–770).

[Com94] Adriana B. Compagnoni. Decidability of higher-order subtyping with intersection types. In *Computer Science Logic*, September 1994. Kazimierz, Poland. Springer *Lecture Notes in Computer Science* 933, June 1995. Also available as University of Edinburgh, LFCS technical report ECS-LFCS-94-281, titled "Subtyping in $F^{\omega}_{\wedge}$ is decidable".

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

[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.

[GP97] Giorgio Ghelli and Benjamin Pierce. Bounded existentials and minimal typing. *Theoretical Computer Science*, 1997. To appear.

[HP95] Martin Hofmann and Benjamin Pierce. A unifying type-theoretic framework for objects. *Journal of Functional Programming*, 5(4):593–635, October 1995. Previous versions appeared in the Symposium on Theoretical Aspects of Computer Science, 1994, (pages 251–262) and, under the title "An Abstract View of Objects and Subtyping (Preliminary Report)," as University of Edinburgh, LFCS technical report ECS-LFCS-92-226, 1992.

[MP88] John Mitchell and Gordon Plotkin. Abstract types have existential type. *ACM Transactions on Programming Languages and Systems*, 10(3), July 1988.

[Pie94] Benjamin C. Pierce. Bounded quantification is undecidable. *Information and Computation*, 112(1):131–165, July 1994. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press, 1994). A preliminary version appeared in POPL '92.

[PS94] Benjamin Pierce and Martin Steffen. Higher-order subtyping. In *IFIP Working Conference on Programming Concepts, Methods and Calculi (PROCOMET)*, 1994. Full version in *Theoretical Computer Science*, vol. 176, no. 1–2, pp. 235–282, 1997.

[PT94] Benjamin C. Pierce and David N. Turner. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming*, 4(2):207–247, April 1994. 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".

[PT97] Benjamin C. Pierce and David N. Turner. Local type argument synthesis with bounded quantification. Technical Report 495, Computer Science Department, Indiana University, 1997.

[PT98] Benjamin C. Pierce and David N. Turner. Local type inference. In *Conference Record of POPL '98: the 25th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1998. To appear. Full version available as Indiana University CSCI technical report #493.