Under consideration for publication in Math. Struct. in Comp. Science

Polarized Subtyping for Sized Types

ANDREAS ABEL[†]
Department of Computer Science, University of Munich
Oettingenstr. 67, D-80538 München, Germany
abel@tcs.ifi.lmu.de

Received March 15, 2006

We present an algorithm for deciding polarized higher-order subtyping without bounded quantification. Constructors are identified not only modulo β, but also η. We give a direct proof of completeness, without constructing a model or establishing a strong normalization theorem. Inductive and coinductive types are enriched with a notion of size and the subtyping calculus is extended to account for the arising inclusions between the sized types.

1. Introduction

Polarized kinding and subtyping has recently received interest in two contexts. First, in the analysis of container types in object-oriented programming languages (Duggan and Compagnoni, 1999): If $\mathsf{List}\,A$ is a functional (meaning: read-only) collection of objects of type A and A is a subtype (subclass) of B then $\mathsf{List}\,A$ should be a subtype of $\mathsf{List}\,B$. However, for read-write collections, as for instance Array, such a subtyping relation is unsound[‡], hence these two collection constructors must be kept apart. The conventional modeling language for object types, System $\mathsf{F}^{\omega}_{\leq}$, does not distinguish List and Array in their kind—both map types to types, thus, have kind $* \to *$. To store subtyping properties in the kind of constructors, polarities were added by Cardelli, Pierce (unpublished), Steffen (1998), and Duggan and Compagnoni (1999). Now, the type constructor List gets kind $* \xrightarrow{+} *$, meaning that it is a monotone (or covariant) type-valued function, whereas Array gets kind $* \xrightarrow{\circ} *$, meaning that Array is neither co- nor contravariant or its variance is unknown to the type system.

Another application of polarized kinding are normalizing languages[§] with recursive datatypes. It is well-known that if a data type definition has a negative recursive occurrence, a looping term can be constructed by just using the constructors and destructors of this data type, without actually requiring recursion on the level of programs (Mendler, 1987). Negative occurrences can be excluded by polarized kinding (Abel and Matthes, 2004)—a recursive type μF is only admitted if $F : * \xrightarrow{+} *$.

A promising way to formulate a normalizing language is by using *sized types*. Hughes, Pareto,

[†] Research supported by the coordination action *TYPES* (510996) and thematic network *Applied Semantics II* (IST-2001-38957) of the European Union and the project *Cover* of the Swedish Foundation of Strategic Research (SSF).

[‡] Nevertheless, such a subtyping rule has been added for arrays in Java.

[§] In a normalizing language, each program is terminating.

and Sabry (1996) have presented such a language, which can be used, e. g., as a basis for embedded programming. It features sized first-order parametric data types, where the size parameter induces a natural subtyping relation. Independently, Barthe et. al. (2004) have arrived at a similar system, which is intended as the core of a theorem prover language. Both systems, however, fail to treat higher-order and heterogeneous (or nested) data types which have received growing interest in the functional programming community (Altenkirch and Reus, 1999; Bird and Paterson, 1999; Hinze, 2000; Okasaki, 1999; Abel et al., 2005).

In order to extend the sized type system to such higher-order constructions, we need to handle polarized higher-order subtyping! Steffen (1998) has already defined the necessary concepts and an algorithm that decides this kind of subtyping. But because he features also bounded quantification, his completeness proof for the algorithm is long and complicated. In this article, I present a different subtyping algorithm, without bounded quantification, but instead fitted to the needs of sized types, and prove it sound and complete in a rather straight-forward manner.

Main technical contribution. We define a polarized higher-order subtyping algorithm that respects not only β but also η-equality and computes the normal form of the considered type constructors incrementally. A novelty is the succinct and direct proof of completeness, which relies neither on a normalization theorem nor a model construction. Instead, a lexicographic induction on kinds and derivations is used.

Organization. In Section 2, we recapitulate the polarized version of F^ω defined in a previous paper (Abel and Matthes, 2004) and extend it by subkinding and polarized higher-order subtyping. A subtyping algorithm is presented in Section 3. In Section 4, we prove completeness of the algorithmic equality. The extension to sized types is presented in Section 5, and we close with a discussion of related work.

Preliminaries. The reader should be familiar with higher-order polymorphism and subtyping. Pierce (2002) provides an excellent introduction.

Judgements. In the following, we summarize the inductively defined judgements used in this article.

$\kappa \leq \kappa'$	κ is a subkind of κ'
$\Gamma \vdash F : \kappa$	constructor F has kind κ in context Γ
$\Gamma \vdash F = F' : \kappa$	F and F' of kind κ are $\beta\eta$-equal
$\Gamma \vdash F \leq F' : \kappa$	F is a higher-order subtype of F'
$F \searrow W$	F has weak head normal form W
$\Gamma \vdash N \leq^q N' \rightrightarrows \kappa$	algorithmic subtyping (inference mode)
$\Gamma \vdash W \leq^q W' \leftleftarrows \kappa$	algorithmic subtyping (checking mode)

When we write $\mathcal{D} :: J$, we mean that judgement J has derivation $\mathcal{D}$. Then, $|\mathcal{D}|$ denotes the height of this derivation. We consider derivations ordered by their height: $\mathcal{D}_1 \leq \mathcal{D}_2$ iff $|\mathcal{D}_1| \leq |\mathcal{D}_2|$.

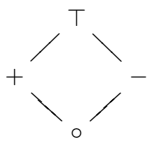
2. Polarized System F^ω

In this section, we present a modification of F^ω where function kinds are decorated with polarities. This is essentially Fix^ω (Abel and Matthes, 2004) without fixed-points, but the additional polarity $\top$, subkinding and subtyping. A technical difference is that Fix^ω uses Church-style (kind-annotated) constructors whereas we use Curry-style (domain-free) constructors. However, all result of this paper apply also to the Church style.

$p, q \in \mathsf{Pol}$

$p, q \quad ::= \quad \circ$ non-variant
$\quad\quad\quad\; \mid \; +$ covariant
$\quad\quad\quad\; \mid \; -$ contravariant
$\quad\quad\quad\; \mid \; \top$ invariant

$p \leq q$



pq

	$\circ$	$+$	$-$	$\top$
$\circ$	$\circ$	$\circ$	$\circ$	$\top$
$+$	$\circ$	$+$	$-$	$\top$
$-$	$\circ$	$-$	$+$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$

Fig. 1. Polarities: definition, ordering, composition.

2.1. *Polarities*

We aim to distinguish constructors with regard to their *monotonicity* or *variance*. For instance, the product constructor $\times$ is *monotone* or *covariant* in both of its arguments. If one enlarges the type A or B, more terms inhabit $A \times B$. The opposite behavior is called *antitone* or *contravariant*. Two more scenarios are possible: the value FA does not change when we modify A. Then F is called *constant* or *invariant*. Finally, a function F might not exhibit a uniform behavior, it might grow or shrink with its argument, or we just do not know how F behaves. This is the general case, we call it *mixed-variant*. Each of the behaviors is called a *polarity* and abbreviated by one of the four symbols displayed in Fig. 1.

The polarities are related: Since *mixed-variant* means that we do not have any information about the function, and we can always disregard our knowledge about variance, each function is mixed-variant. The inclusion order between the four sets of in-, co-, contra-, and mixed-variant functions induces a partial *information order* $\leq$ on Pol. The smaller a set is, the more information it carries. Hence $\circ \leq p$, $p \leq \top$, and $p \leq p$ for all p. This makes Pol a bounded 4-element lattice as visualized in Fig. 1.

Polarity of composed functions. Let F, G be two functions such that the composition $F \circ G$ is well-defined. If F has polarity p and G has polarity q, we denote the polarity of the composed function $F \circ G$ by pq. It is clear that polarity composition is monotone: if one gets more information about F or G, certainly one cannot have *less* information about $F \circ G$. Then, if one of the functions is constant, so is their composition. Otherwise, if one of them is mixed-variant, the same holds for the composition. In the remaining cases, the composition is covariant if F and G have the same variance, otherwise it is contravariant. This yields the multiplication table in

Fig. 1. Polarity composition, as function composition, is associative. It is even commutative, but not *a priori*, since function composition is not commutative.

Inverse application of polarities. If $f(y) = py$ is the function which composes a polarity with p, what would be its inverse $g(x) = p^{-1}x$? It is possible to define g in such a way that f and g form a Galois connection, i.e.,

$$p^{-1}x \leq y \iff x \leq py.$$

It is not hard to see that the unique solution is given by the equations: $+^{-1}x = x$, $-^{-1}x = -x$, $\top^{-1}x = \circ$, $\circ^{-1}\circ = \circ$ and $\circ^{-1}x' = \top$ (for $x' \neq \circ$). As for every Galois connection, it holds that $p^{-1}py \leq y$ and $x \leq pp^{-1}x$, and both f and g are monotone.

2.2. *Kinds*

Constructors are classified by their *kind*, i. e., as types, functions on types, functions on such functions etc.

$$\begin{array}{rcll} \mathsf{Kind} \ni \kappa & ::= & * & \text{types} \\ & \mid & p\kappa_1 \to \kappa_2 & p\text{-variant constructor transformers} \end{array}$$

A constructor of kind $p\kappa_1 \to \kappa_2$ is a p-variant function which maps constructors of kind κ_1 to constructors of kind κ_2. Sometimes it is written as $\kappa_1 \xrightarrow{p} \kappa_2$. Let $\vec{p}\vec{\kappa} \to \kappa'$ be an abbreviation for $p_1\kappa_1 \to \cdots \to p_n\kappa_n \to \kappa'$ where we presuppose $|\vec{p}| = |\vec{\kappa}| = n$. It is clear that every kind can be written as $\vec{p}\vec{\kappa} \to *$ with potentially empty vectors $\vec{p}$ and $\vec{\kappa}$. The *rank* of a kind is defined recursively as $\mathsf{rk}(\vec{p}\vec{\kappa} \to *) = \max\{1 + \mathsf{rk}(\kappa_i) \mid 1 \leq i \leq |\vec{\kappa}|\}$ (where the maximum of an empty set is 0).

Subkinding. The order on polarities induces an order $\kappa \leq \kappa'$ on kinds. We say that κ *is a subkind of* κ' or κ' *is a superkind of* κ. As usual, this shall mean that each constructor of kind κ is also of kind κ'. The subkinding relation is given inductively by the following rules:

$$\frac{}{* \leq *} \qquad \frac{}{\mathsf{ord} \leq \mathsf{ord}} \qquad \frac{p' \leq p \qquad \kappa_1' \leq \kappa_1 \qquad \kappa_2 \leq \kappa_2'}{p\kappa_1 \to \kappa_2 \leq p'\kappa_1' \to \kappa_2'}$$

Subkinding is reflexive, transitive, and antisymmetric; hence, a proper partial order.

Lemma 2.1 (Inversion of subkinding). Related kinds have the same shape:

1 If $\vec{p}\vec{\kappa} \to \kappa_0 \leq \kappa'$ then there are $\vec{p}', \vec{\kappa}', \kappa_0'$ with $\kappa' = \vec{p}'\vec{\kappa}' \to \kappa_0'$ and $|\vec{p}'| = |\vec{\kappa}'| = |\vec{p}| = |\vec{\kappa}| =: n$ such that $\kappa_0 \leq \kappa_0'$ and both $p_i' \leq p_i$ and $\kappa_i' \leq \kappa_i$ for all $1 \leq i \leq n$.
2 The last sentence remains true when we trade $\leq$ for $\geq$.

2.3. *Constructors*

Constructors are given by the following Curry-style type-level lambda-calculus with some constants. The meta-variable X ranges over a countably infinite set of constructor variables and C over a fixed set of constructor constants.

$$\mathsf{Constr} \ni A, B, F, G, H, I, J ::= C \mid X \mid \lambda X F \mid F\,G$$

As usual, $\lambda X F$ binds variable X in F. We identify constructors under α-equivalence, i. e., under renaming of bound variables. $\mathsf{FV}(F)$ shall denote the set of free variables of constructor F.

Signature. The constructor constants C are taken from a fixed *signature* Σ which contains at least the following constants together with their kinding:

$$\begin{array}{rll} & \to \;:\; * \xrightarrow{-} * \xrightarrow{+} * & \text{function space,} \\ \text{for each } \kappa\text{:} & \forall_\kappa \;:\; (\kappa \xrightarrow{\circ} *) \xrightarrow{+} * & \text{quantification.} \end{array}$$

2.4. *Kinding*

In this section, we present rules of *kinding*, i. e., assigning kinds to constructors. The rules extend the kinding rules of F^ω by the treatment of polarities.

Polarized contexts. A polarized context Γ fixes a polarity p and a kind κ for each free variable X of a constructor F. If $p = +$, then X may only appear positively in F; this ensures that $\lambda X F$ is an monotone function. Similarly, if $p = -$, then X may only occur negatively, and if $p = \circ$, then X may appear in both positive and negative positions. A variable labeled with $\top$ may only appear in arguments of an invariant function.

$$\begin{array}{rcll} \mathsf{PCxt} \ni \Gamma & ::= & \diamond & \text{empty context} \\ & \mid & \Gamma,\, X{:}p\kappa & \text{extended context } (X \notin \mathsf{dom}(\Gamma)) \end{array}$$

The domain $\mathsf{dom}(\Gamma)$ is the set of constructor variables Γ mentions. As usual, each variable can appear in the context only once. The order of hypotheses $X : p\kappa$ in the context does not matter, so we may silently reorder them.

Ordering on contexts. We say context Γ' is more *liberal* than context Γ, written $\Gamma' \le \Gamma$, iff for all variables X,

$$(X{:}p\kappa) \in \Gamma \text{ implies } (X{:}p'\kappa') \in \Gamma' \text{ for some } p' \le p \text{ and } \kappa' \le \kappa.$$

In particular, Γ' may declare more variables than Γ and assign weaker polarities to them. The intuition is that all constructors which are well-kinded in Γ are also well-kinded in a more permissive context Γ'.

Application of polarities to contexts. The application $p\Gamma$ of a polarity p to a context Γ is defined as pointwise application, i. e., if $(X : q\kappa) \in \Gamma$, then $(X : (pq)\kappa) \in p\Gamma$. Inverse application $p^{-1}\Gamma$ is defined analogously. Together, they form a Galois connection, i. e., for all Γ and Γ',

$$p^{-1}\Gamma \le \Gamma' \iff \Gamma \le p\Gamma'.$$

Lemma 2.2. If $\Gamma_1 \le \Gamma_2$ and $q_1 \le q_2$ then $q_1\Gamma_1 \le q_2\Gamma_2$ and $q_2^{-1}\Gamma_1 \le q_1^{-1}\Gamma_2$.

Lemma 2.3 (Some simplification laws). $p(q\Gamma) = (pq)\Gamma$, $p^{-1}(q\Gamma) = (p^{-1}q)\Gamma$, $+^{-1}\Gamma = +\Gamma = \Gamma$, $-^{-1}\Gamma = -\Gamma$, $--\Gamma = \Gamma$.

Kinding. We will introduce a judgement $\Gamma \vdash F : \kappa$ which combines the usual notions of well-kindedness and positive and negative occurrences of type variables. A candidate for the application rule is

$$\frac{\Gamma \vdash F : p\kappa \to \kappa' \qquad \Gamma' \vdash G : \kappa}{\Gamma \vdash F\,G : \kappa'}\ \Gamma \leq p\Gamma'.$$

The side condition is motivated by polarity composition. Consider the case that $X \notin \mathsf{FV}(F)$. If G is viewed as a function of X, then $F\,G$ is the composition of F and G. Now if G is q-variant in X, then $F\,G$ is pq-variant in X. This means that all q-variant variables of Γ' must appear in Γ with a polarity of at most pq. Now if $X \in \mathsf{FV}(F)$, it could be that it is actually declared in Γ with a polarity smaller than pq. Also, variables which are not free in G are not affected by the application $F\,G$, hence they can carry the same polarity in $F\,G$ as in F. Together this motivates the condition $\Gamma \leq p\Gamma'$.

Since $p^{-1}\Gamma$ is the most liberal context which satisfies the side condition, we can safely replace Γ' by $p^{-1}\Gamma$ in the above rule. Hence, we arrive at the following formulation of the kinding rules:

$$\textsc{kind-c}\ \frac{C : \kappa \in \Sigma}{\Gamma \vdash C : \kappa} \qquad \textsc{kind-var}\ \frac{X : p\kappa \in \Gamma \qquad p \leq +}{\Gamma \vdash X : \kappa}$$

$$\textsc{kind-}\lambda\ \frac{\Gamma, X : p\kappa \vdash F : \kappa'}{\Gamma \vdash \lambda X F : p\kappa \to \kappa'} \qquad \textsc{kind-app}\ \frac{\Gamma \vdash F : p\kappa \to \kappa' \qquad p^{-1}\Gamma \vdash G : \kappa}{\Gamma \vdash F\,G : \kappa'}$$

$$\textsc{kind-sub}\ \frac{\Gamma \vdash F : \kappa \qquad \kappa \leq \kappa'}{\Gamma \vdash F : \kappa'}$$

Although these rules are not fully deterministic, they can easily be turned into a bidirectional kind checking algorithm for constructors in β-normal form (see, e. g., Davies and Pfenning (2000)).

Example 2.4 (Derived rules for function space and quantification). The following rules are derivable:

$$\frac{-\Gamma \vdash A : * \qquad \Gamma \vdash B : *}{\Gamma \vdash A \to B : *} \qquad \frac{\Gamma, X : \circ\kappa \vdash A : *}{\Gamma \vdash \forall_\kappa \lambda X A : *}$$

Lemma 2.5 (Admissible rules for kinding).

1 Weakening: If $\mathcal{D} :: \Gamma \vdash F : \kappa$ and both $\Gamma' \leq \Gamma$ and $\kappa \leq \kappa'$, then $\mathcal{D}' :: \Gamma' \vdash F : \kappa'$ for some $\mathcal{D}' \leq \mathcal{D}$.

2 Strengthening: If $\mathcal{D} :: \Gamma, X : p\kappa \vdash F : \kappa'$ and $X \notin \mathsf{FV}(F)$, then $\mathcal{D}' :: \Gamma \vdash F : \kappa'$ for some $\mathcal{D}' \leq \mathcal{D}$.

3 Substitution: If $\mathcal{D} :: \Gamma, X : p\kappa \vdash F : \kappa'$ and $p^{-1}\Gamma \vdash G : \kappa$, then $\Gamma \vdash [G/X]F : \kappa'$.

Proof. Each by induction on $\mathcal{D}$. □

Lemma 2.6 (Inversion of kinding).

1 If $\mathcal{D} :: \Gamma \vdash C : \kappa'$ then $(C : \kappa) \in \Sigma$ for some $\kappa \leq \kappa'$.

2 If $\mathcal{D} :: \Gamma \vdash X : \kappa'$ then $(X : p\kappa) \in \Gamma$ for some $p \leq +$, $\kappa \leq \kappa'$.

3 If $\mathcal{D} :: \Gamma \vdash \lambda X F : \kappa'$ then $\kappa' = p\kappa_1 \to \kappa_2$ for some p, κ_1, κ_2 and $\mathcal{D}' :: \Gamma, X : p\kappa_1 \vdash F : \kappa_2$ for some $\mathcal{D}' < \mathcal{D}$.

4 If $\mathcal{D} :: \Gamma \vdash F\,G : \kappa'$ then $\mathcal{D}_1 :: \Gamma \vdash F : p\kappa \to \kappa'$ and $\mathcal{D}_2 :: p^{-1}\Gamma \vdash G : \kappa$ for some p, κ, and $\mathcal{D}_1, \mathcal{D}_2 < \mathcal{D}$.

Proof. Each by induction on $\mathcal{D}$. For example, 3: If the last rule in $\mathcal{D}$ was KIND-λ, we are done, otherwise, the last rule must have been KIND-SUB:

$$\frac{\Gamma \vdash \lambda X F : \kappa \qquad \kappa \leq \kappa'}{\Gamma \vdash \lambda X F : \kappa'}$$

By induction hypothesis, $\kappa = p\kappa_1 \to \kappa_2$ and $\Gamma, X : p\kappa_1 \vdash F : \kappa_2$. By inversion on subkinding, $\kappa' = p'\kappa_1' \to \kappa_2'$ with $p' \leq p$, $\kappa_1' \leq \kappa_1$, and $\kappa_2 \leq \kappa_2'$. This entails that the context $(\Gamma, X : p'\kappa_1')$ is more liberal than $(\Gamma, X : p\kappa_1)$, hence, we can apply the weakening lemma to obtain $\Gamma, X : p'\kappa_1' \vdash F : \kappa_2'$. □

2.5. *Equality*

In contrast to most presentations of System F^ω, we consider constructors equivalent modulo β *and* η.

Axioms.

$$\text{EQ-}\beta\ \frac{\Gamma, X : p\kappa \vdash F : \kappa' \qquad p^{-1}\Gamma \vdash G : \kappa}{\Gamma \vdash (\lambda X F)\, G = [G/X]F : \kappa'}$$

$$\text{EQ-}\eta\ \frac{\Gamma \vdash F : p\kappa \to \kappa'}{\Gamma \vdash (\lambda X.\, FX) = F : p\kappa \to \kappa'}\ X \notin \mathsf{FV}(F)$$

$$\text{EQ-}\top\ \frac{\Gamma \vdash F : \top\kappa \to \kappa' \qquad \top^{-1}\Gamma \vdash G : \kappa \qquad \top^{-1}\Gamma \vdash G' : \kappa}{\Gamma \vdash F\, G = F\, G' : \kappa'}$$

Rule EQ-$\top$ expresses that the value of an invariant function does not depend on its argument. Only well-kindedness of the argument is required, in order to ensure the validity property (Lemma 2.7.1).

Congruence rules and subsumption.

$$\text{EQ-C}\ \frac{C : \kappa \in \Sigma}{\Gamma \vdash C = C : \kappa} \qquad \text{EQ-VAR}\ \frac{X : p\kappa \in \Gamma \qquad p \leq +}{\Gamma \vdash X = X : \kappa}$$

$$\text{EQ-}\lambda\ \frac{\Gamma, X : p\kappa \vdash F = F' : \kappa'}{\Gamma \vdash \lambda X F = \lambda X F' : p\kappa \to \kappa'} \qquad \text{EQ-APP}\ \frac{\Gamma \vdash F = F' : p\kappa \to \kappa' \qquad p^{-1}\Gamma \vdash G = G' : \kappa}{\Gamma \vdash F\, G = F'\, G' : \kappa'}$$

$$\text{EQ-SUB}\ \frac{\Gamma \vdash F = F' : \kappa \qquad \kappa \leq \kappa'}{\Gamma \vdash F = F' : \kappa'}$$

Symmetry and transitivity.

$$\text{EQ-SYM}\ \frac{\Gamma \vdash F = F' : \kappa}{\Gamma \vdash F' = F : \kappa} \qquad \text{EQ-TRANS}\ \frac{\Gamma \vdash F_1 = F_2 : \kappa \qquad \Gamma \vdash F_2 = F_3 : \kappa}{\Gamma \vdash F_1 = F_3 : \kappa}$$

2.6. *Subtyping*

In this section, we specify subtyping for constructors of polarized kinds. The rules are inspired by Steffen (1998).

Reflexivity, transitivity and antisymmetry. These three properties make subtyping a partial order on constructors of the same kind.

$$\text{LEQ-REFL}\ \frac{\Gamma \vdash F = F' : \kappa}{\Gamma \vdash F \leq F' : \kappa} \qquad \text{LEQ-TRANS}\ \frac{\Gamma \vdash F_1 \leq F_2 : \kappa \qquad \Gamma \vdash F_2 \leq F_3 : \kappa}{\Gamma \vdash F_1 \leq F_3 : \kappa}$$

$$\text{LEQ-ANTISYM}\ \frac{\Gamma \vdash F \leq F' : \kappa \qquad \Gamma \vdash F' \leq F : \kappa}{\Gamma \vdash F = F' : \kappa}$$

The reflexivity rule LEQ-REFL includes the subtyping axioms for variables and constants as special cases. Reflexivity and transitivity together ensure that subtyping is compatible with equality. The antisymmetry rule LEQ-ANTISYM potentially enlarges our notion of equality.

Abstraction.

$$\text{LEQ-}\lambda\ \frac{\Gamma,\, X : p\kappa \vdash F \leq F' : \kappa'}{\Gamma \vdash \lambda X F \leq \lambda X F' : p\kappa \to \kappa'}$$

Application. There are two kinds of congruence rules for application: one kind states that if functions F and F' are in the subtyping relation, so are their values $F\,G$ and $F'\,G$ at a certain argument G.

$$\text{LEQ-FUN}\ \frac{\Gamma \vdash F \leq F' : p\kappa \to \kappa' \qquad p^{-1}\Gamma \vdash G : \kappa}{\Gamma \vdash F\,G \leq F'\,G : \kappa'}$$

The other kind of rules concern the opposite case: If F is a function and two arguments G and G' are in a subtyping relation, so are the values $F\,G$ and $F\,G'$ of the function at these arguments. However, such a relation can only exist if F is covariant or contravariant.

$$\text{LEQ-ARG}+\ \frac{\Gamma \vdash F : +\kappa \to \kappa' \qquad \Gamma \vdash G \leq G' : \kappa}{\Gamma \vdash F\,G \leq F\,G' : \kappa'}$$

$$\text{LEQ-ARG}-\ \frac{\Gamma \vdash F : -\kappa \to \kappa' \qquad -\Gamma \vdash G' \leq G : \kappa}{\Gamma \vdash F\,G \leq F\,G' : \kappa'}$$

Note that we only included a subsumption rule for equality (EQ-SUB), not for subtyping. Subsumption for subtyping is admissible, as item 2 of the following lemma states:

Lemma 2.7 (Admissible rules for equality and subtyping I). Let $\mathcal{R} \in \{=, \leq\}$.

1 Validity: If $\mathcal{D} :: \Gamma \vdash F\ \mathcal{R}\ F' : \kappa$ then $\Gamma \vdash F : \kappa$ and $\Gamma \vdash F' : \kappa$.
2 Weakening: If $\mathcal{D} :: \Gamma \vdash F\ \mathcal{R}\ F' : \kappa$ and both $\Gamma' \leq \Gamma$ and $\kappa \leq \kappa'$, then $\Gamma' \vdash F\ \mathcal{R}\ F' : \kappa'$.
3 Strengthening: If $\mathcal{D} :: \Gamma, X : p\kappa \vdash F\ \mathcal{R}\ F' : \kappa'$ and $X \notin \mathsf{FV}(F, F')$, then $\Gamma \vdash F\ \mathcal{R}\ F' : \kappa'$.

Proof. Each by induction on $\mathcal{D}$. □

Lemma 2.8 (Admissible rules for equality and subtyping II). See Fig. 2.

3. Algorithmic Polarized Subtyping

In this section, we present an algorithm for deciding whether two well-kinded constructors are equal or related by subtyping. The algorithm is an adaption of Coquand's $\beta\eta$-equality test (Coquand, 1991) to the needs of subtyping and polarities. The idea is to first weak-head normalize

$$\textsc{eq-refl}\ \frac{\Gamma \vdash F : \kappa}{\Gamma \vdash F = F : \kappa}$$

$$\textsc{leq-app}\circ\ \frac{\Gamma \vdash F \leq F' : \circ\kappa \to \kappa' \qquad \circ^{-1}\Gamma \vdash G = G' : \kappa}{\Gamma \vdash F\,G \leq F'\,G' : \kappa'}$$

$$\textsc{leq-app}+\ \frac{\Gamma \vdash F \leq F' : +\kappa \to \kappa' \qquad \Gamma \vdash G \leq G' : \kappa}{\Gamma \vdash F\,G \leq F'\,G' : \kappa'}$$

$$\textsc{leq-app}-\ \frac{\Gamma \vdash F \leq F' : -\kappa \to \kappa' \qquad -\Gamma \vdash G' \leq G : \kappa}{\Gamma \vdash F\,G \leq F'\,G' : \kappa'}$$

$$\textsc{leq-app}\top\ \frac{\Gamma \vdash F \leq F' : \top\kappa \to \kappa' \qquad \top^{-1}\Gamma \vdash G : \kappa \qquad \top^{-1}\Gamma \vdash G' : \kappa}{\Gamma \vdash F\,G \leq F'\,G' : \kappa'}$$

$$\textsc{eq-app-}\top\ \frac{\Gamma \vdash F = F' : \top\kappa \to \kappa' \qquad \top^{-1}\Gamma \vdash G : \kappa \qquad \top^{-1}\Gamma \vdash G' : \kappa}{\Gamma \vdash F\,G = F'\,G' : \kappa'}$$

Fig. 2. Admissible rules for equality and subtyping.

the constructors under consideration and then compare their head symbols. If they are related, one continues to recursively compare the subcomponents, otherwise subtyping fails.

Weak head normal forms $W \in \mathsf{Val}$ are given by the grammar:

$$\begin{array}{llll} \mathsf{Ne} & \ni & N & ::= \quad C \mid X \mid N\,G \qquad \text{neutral constructors} \\ \mathsf{Val} & \ni & V, W & ::= \quad N \mid \lambda X F \qquad \text{weak head values} \end{array}$$

Weak head evaluation $F \searrow W$, a big-step call-by-name operational semantics, is defined inductively by the following rules:

$$\textsc{eval-c}\ \frac{}{C \searrow C} \qquad \textsc{eval-var}\ \frac{}{X \searrow X} \qquad \textsc{eval-lam}\ \frac{}{\lambda X F \searrow \lambda X F}$$

$$\textsc{eval-app-ne}\ \frac{F \searrow N}{F\,G \searrow N\,G} \qquad \textsc{eval-app-}\beta\ \frac{F \searrow \lambda X F' \qquad [G/X]F' \searrow W}{F\,G \searrow W}$$

Lemma 3.1 (Properties of weak head evaluation).

1 If $F\,X \searrow W$ then either $F \searrow \lambda X F'$ and $F' \searrow W$, or $F \searrow N$ and $W = N\,X$. If $X \notin \mathsf{FV}(F)$ then $X \notin \mathsf{FV}(N)$.
2 If $F \searrow W$ and $[G/X]W \searrow V$ then $[G/X]F \searrow V$.
3 If $F \searrow W$ and $W\,G \searrow V$ then $F\,G \searrow V$.

Lemma 3.2 (Soundness of weak head evaluation). If $\Gamma \vdash F : \kappa$ and $F \searrow W$ then $\Gamma \vdash F = W : \kappa$.

Proof. By induction on $F \searrow W$. The interesting case is:

$$\textsc{eval-app-}\beta\ \frac{F \searrow \lambda X F' \qquad [G/X]F' \searrow W}{F\,G \searrow W}$$

$\Gamma \vdash F\,G : \kappa'$	by assumption
$\Gamma \vdash F : p\kappa \to \kappa'$	and
$p^{-1}\Gamma \vdash G : \kappa$	by inversion
$\Gamma \vdash F = \lambda X F' : p\kappa \to \kappa'$	by induction hypothesis
$\Gamma \vdash \lambda X F' : p\kappa \to \kappa'$	by validity
$\Gamma, X : p\kappa \vdash F' : \kappa'$	by inversion
$\Gamma \vdash (\lambda X F')\,G = [G/X]F' : \kappa'$	by rule EQ-β
$\Gamma \vdash [G/X]F' : \kappa'$	by substitution
$\Gamma \vdash [G/X]F' = W : \kappa'$	by induction hypothesis
$p^{-1}\Gamma \vdash G = G : \kappa$	by reflexivity
$\Gamma \vdash F\,G = (\lambda X F')\,G : \kappa'$	by rule EQ-APP
$\Gamma \vdash F\,G = W : \kappa'$	by rule EQ-TRANS

□

Corollary 3.3 (Subject reduction). If $\Gamma \vdash F : \kappa$ and $F \searrow W$ then $\Gamma \vdash W : \kappa$.

Proof. From the lemma by validity (Lemma 2.7.1). □

We are ready to define the subtyping algorithm. Note that at any point during subtyping checking we may require kinding information. For example, consider checking $X\,G \leq X\,G'$. If X is covariant, we need to continue with $G \leq G'$, but if X is contravariant, the next step would be checking $G' \leq G$. Hence, the algorithm needs both context Γ and kind κ of the two considered constructors as additional input. Although the subtyping algorithm is kinded, it is not *kind-directed* as, for instance, Harper and Pfenning's (2005) algorithmic equality for the logical framework LF.

The general form of the algorithmic subtyping judgement is defined by

$$\begin{aligned}\Gamma \vdash F \leq^q F' \leftleftarrows \kappa \quad :\Longleftrightarrow \quad & q = \top \\ \text{or} \quad & F \searrow W \text{ and } F' \searrow W' \text{ and } \Gamma \vdash W \leq^q W' \leftleftarrows \kappa,\end{aligned}$$

where $\Gamma \vdash W \leq^q W' \leftleftarrows \kappa$ is a judgement defined inductively in Fig. 3. The polarity q codes the relation that we seek to establish between F and F': If $q = \circ$, we expect them to be equal, if $q = +$, we expect $F \leq F'$, and if $q = -$, then the other way round. Finally if $q = \top$, then F and F' need not be related, and the algorithm succeeds immediately.

The judgements for algorithmic subtyping $\Gamma \vdash N \leq^q N' \rightrightarrows \kappa$ for neutral constructors and $\Gamma \vdash W \leq^q W' \leftleftarrows \kappa$ for weak head values are defined inductively by the rules in Fig. 3. They are deterministic and can be directly implemented as an algorithm (apply the rules backwards). Both judgements take the context Γ, the polarity q, and the two constructors as input. Judgement $\Gamma \vdash N \leq^q N' \rightrightarrows \kappa$ produces kind κ if it succeeds, whereas the judgement $\Gamma \vdash F \leq^q F' \leftleftarrows \kappa$

$$\text{AL-C}\ \frac{(C:\kappa)\in\Sigma}{\Gamma\vdash C\leq^q C\rightrightarrows\kappa}\qquad \text{AL-VAR}\ \frac{(X:p\kappa)\in\Gamma\qquad p\leq +}{\Gamma\vdash X\leq^q X\rightrightarrows\kappa}$$

$$\text{AL-APP-NE}\ \frac{\Gamma\vdash N\leq^q N'\rightrightarrows p\kappa\to\kappa'\qquad p^{-1}\Gamma\vdash G\leq^{pq} G'\leftrightarrows\kappa}{\Gamma\vdash N\,G\leq^q N'\,G'\rightrightarrows\kappa'}$$

$$\text{AL-NE}\ \frac{\Gamma\vdash N\leq^q N'\rightrightarrows\kappa'}{\Gamma\vdash N\leq^q N'\leftrightarrows\kappa}$$

$$\text{AL-}\lambda\ \frac{\Gamma,X:p\kappa\vdash F\leq^q F'\leftrightarrows\kappa'}{\Gamma\vdash\lambda XF\leq^q\lambda XF'\leftrightarrows p\kappa\to\kappa'}$$

$$\text{For } X\notin\mathsf{FV}(N):\quad \text{AL-}\eta\text{-L}\ \frac{\Gamma,X:p\kappa\vdash F\leq^q N\,X:\kappa'}{\Gamma\vdash\lambda XF\leq^q N:p\kappa\to\kappa'}\quad \text{AL-}\eta\text{-R}\ \frac{\Gamma,X:p\kappa\vdash N\,X\leq^q F:\kappa'}{\Gamma\vdash N\leq^q\lambda XF:p\kappa\to\kappa'}$$

Fig. 3. Algorithmic subtyping and equality.

takes κ as an additional input and either succeeds or fails. The direction of the double arrow indicates the flow of the kinding information out of ($\leftrightarrows$) or into ($\rightrightarrows$) κ.

Note that due to rule AL-NE, algorithmic subtyping does not guarantee that the involved constructors have the ascribed kind. For example, $X:+*\vdash X\leq^\circ X\leftrightarrows -*\to *$ is a valid derivation. It does not even entail that the involved constructors are well-kinded at all. For example:

$$Y:+(*\xrightarrow{+}*),X:+(*\xrightarrow{\circ}*)\vdash YX\leq^\circ YX\rightrightarrows *$$

Consequently, algorithmic subtyping is only sound for well-kinded constructors:

Theorem 3.4 (Soundness of algorithmic subtyping). Let $\Gamma\vdash N,N',W,W':\kappa$.

1 If $\mathcal{D}::\Gamma\vdash N\leq^\circ N'\rightrightarrows\kappa'$ then $\Gamma\vdash N=N':\kappa'$ and $\kappa'\leq\kappa$.
2 If $\mathcal{D}::\Gamma\vdash N\leq^+ N'\rightrightarrows\kappa'$ then $\Gamma\vdash N\leq N':\kappa'$ and $\kappa'\leq\kappa$.
3 If $\mathcal{D}::\Gamma\vdash N\leq^- N'\rightrightarrows\kappa'$ then $\Gamma\vdash N'\leq N:\kappa'$ and $\kappa'\leq\kappa$.
4 If $\mathcal{D}::\Gamma\vdash W\leq^\circ W'\leftrightarrows\kappa$ then $\Gamma\vdash W=W':\kappa$.
5 If $\mathcal{D}::\Gamma\vdash W\leq^+ W'\leftrightarrows\kappa$ then $\Gamma\vdash W\leq W':\kappa$.
6 If $\mathcal{D}::\Gamma\vdash W\leq^- W'\leftrightarrows\kappa$ then $\Gamma\vdash W'\leq W:\kappa$.

Corollary 3.5. The soundness results hold also in the general form $\Gamma\vdash F\leq^q F'\leftrightarrows\kappa$ for well-kinded F, F'.

Proof of the corollary. For example, assume $F\searrow W$, $F'\searrow W'$, and $\Gamma\vdash W\leq^+ W'\leftrightarrows\kappa$. By the theorem $\Gamma\vdash W\leq W':\kappa$, and by soundness of weak head evaluation, $\Gamma\vdash F=W:\kappa$ and similarly for F', W'. Since subtyping is compatible with equality, we get $\Gamma\vdash F\leq F':\kappa$ (to see this, use rules LEQ-REFL and LEQ-TRANS). □

Proof of the soundness theorem. Simultaneously, by induction on $\mathcal{D}$ and by inversion on the typing derivations. We pick some representative cases:

Case

$$\text{AL-APP-NE}\ \frac{\Gamma \vdash N \leq^{+} N' \rightrightarrows p'\kappa_1' \to \kappa_2' \qquad p'^{-1}\Gamma \vdash G \leq^{p'} G' \leftrightarrows \kappa_1'}{\Gamma \vdash N\,G \leq^{+} N'\,G' \rightrightarrows \kappa_2'}$$

$\Gamma \vdash N\,G, N'\,G' : \kappa_2$	by assumption
$\Gamma \vdash N, N' : p\kappa_1 \to \kappa_2$	by inversion
$p^{-1}\Gamma \vdash G, G' : \kappa_1$	by inversion
$\Gamma \vdash N \leq N' : p'\kappa_1' \to \kappa_2'$	by induction hypothesis
$p'\kappa_1' \to \kappa_2' \leq p\kappa_1 \to \kappa_2$	by induction hypothesis
$p \leq p', \kappa_1 \leq \kappa_1', \kappa_2' \leq \kappa_2$	by inversion
$p'^{-1}\Gamma \leq p^{-1}\Gamma$	by Lemma 2.2
$p'^{-1}\Gamma \vdash G, G' : \kappa_1'$	by weakening (Lemma 2.5)

If $p' = \top$, we are immediately done by LEQ-APP$\top$ (Lemma 2.8). Otherwise, obtain the induction hypothesis for G, G' and apply LEQ-APPp.

Case

$$\text{AL-NE}\ \frac{\Gamma \vdash N \leq^{-} N' \rightrightarrows \kappa'}{\Gamma \vdash N \leq^{-} N' \leftrightarrows \kappa}$$

By induction hypothesis $\Gamma \vdash N' \leq N : \kappa'$ and $\kappa' \leq \kappa$. Hence, by weakening, $\Gamma \vdash N' \leq N : \kappa$.

Case

$$\text{AL-}\eta\text{-L}\ \frac{\Gamma, X{:}p\kappa \vdash F \leq^{\circ} N\,X \leftrightarrows \kappa'}{\Gamma \vdash \lambda X F \leq^{\circ} N \leftrightarrows p\kappa \to \kappa'}\ X \notin \mathsf{FV}(N)$$

By induction hypothesis, $\Gamma, X : p\kappa \vdash F = N\,X : \kappa'$, which gives by validity $\Gamma, X : p\kappa \vdash N\,X : \kappa'$. Since $X \notin \mathsf{FV}(N)$, we can infer by inversion and strengthening that $\Gamma \vdash N : p\kappa \to \kappa'$. Rule EQ-$\eta$ entails $\Gamma \vdash \lambda X.\,N\,X = N : p\kappa \to \kappa'$. From the induction hypothesis we also get $\Gamma \vdash \lambda X F = \lambda X.\,N\,X : p\kappa \to \kappa'$ (rule LEQ-λ), hence the goal $\Gamma \vdash \lambda X F = N : p\kappa \to \kappa'$ follows by transitivity.

□

4. Completeness

While soundness of the algorithmic subtyping/equality is easy to show, the opposite direction, completeness, is usually hard and requires either the construction of a model (Compagnoni and Goguen, 2003; Harper and Pfenning, 2005) or strong normalization for constructors (Pierce and Steffen, 1997; Steffen, 1998; Goguen, 2005). We will require neither.

Algorithmic subtyping is *cut-free* in a twofold sense: First, a rule for transitivity is missing (this is the cut on the level of subtyping). Its admissibility can often be shown directly by induction on the derivations (Pierce and Steffen, 1997; Compagnoni and Goguen, 2003)—so also in our case. The second kind of cut is on the level of kinds: Kinds can be viewed as propositions in minimal logic and constructors as their proof terms, and an application which introduces a redex is a cut in natural deduction. Algorithmic subtyping lacks a general rule for application; its admissibility corresponds to the property of normalization or cut admissibility, resp. We manage to show the

admissibility of application directly by a lexicographic induction of the kind of the argument part and the derivation length of the function part. This way, we save ourselves considerable work, and completeness is relatively straightforward.

Lemma 4.1 (Weakening). Let $\Gamma' \leq \Gamma$ and $q \leq q' \neq \top$.

1 If $\mathcal{D} :: \Gamma \vdash N \leq^{q} N' \rightrightarrows \kappa$ then $\mathcal{D}' :: \Gamma' \vdash N \leq^{q'} N' \rightrightarrows \kappa'$ for some $\mathcal{D}' \leq \mathcal{D}$ and $\kappa' \leq \kappa$.
2 If $\mathcal{D} :: \Gamma \vdash W \leq^{q} W' \leftrightarrows \kappa$ and $\kappa \leq \kappa'$ then $\mathcal{D}' :: \Gamma' \vdash W \leq^{q'} W' \leftrightarrows \kappa'$ for some $\mathcal{D}' \leq \mathcal{D}$.

Corollary 4.2 (Weakening). Let $\Gamma' \leq \Gamma$, $q \leq q'$, and $\kappa \leq \kappa'$. If $\mathcal{D} :: \Gamma \vdash F \leq^{q} F' \leftrightarrows \kappa$ then $\mathcal{D}' :: \Gamma' \vdash F \leq^{q'} F' \leftrightarrows \kappa'$ for some $\mathcal{D}' \leq \mathcal{D}$.

Lemma 4.3 (Strengthening). Assume $X \notin \mathsf{FV}(N, N', F, F')$.

1 If $\Gamma, X : p\kappa \vdash N \leq^{q} N' \rightrightarrows \kappa'$ then $\Gamma \vdash N \leq^{q} N' \rightrightarrows \kappa'$.
2 If $\Gamma, X : p\kappa \vdash F \leq^{q} F' \leftrightarrows \kappa'$ then $\Gamma \vdash F \leq^{q} F' \leftrightarrows \kappa'$.

Lemma 4.4 (Swapping).

1 If $\Gamma \vdash N \leq^{q} N' \rightrightarrows \kappa$ then $\Gamma \vdash N' \leq^{-q} N \rightrightarrows \kappa$.
2 If $\Gamma \vdash F \leq^{q} F' \leftrightarrows \kappa$ then $\Gamma \vdash F' \leq^{-q} F \leftrightarrows \kappa$.

Antisymmetry of algorithmic subtyping is straightforward in our case since our judgement is deterministic—it is more difficult in the presence of bounded quantification (Compagnoni and Goguen, 1999).

Lemma 4.5 (Antisymmetry).

1 If $\Gamma \vdash N \leq^{q} N' \rightrightarrows \kappa$ and $\Gamma \vdash N \leq^{q'} N' \rightrightarrows \kappa'$ then $\kappa = \kappa'$ and $\Gamma \vdash N \leq^{\inf(q,q')} N' \rightrightarrows \kappa$.
2 If $\Gamma \vdash F \leq^{q} F' \leftrightarrows \kappa$ and $\Gamma \vdash F \leq^{q'} F' \leftrightarrows \kappa$ then $\Gamma \vdash F \leq^{\inf(q,q')} F' \leftrightarrows \kappa$.

Proof. Simultaneously by induction. The proof is almost trivial, since algorithmic subtyping is deterministic. The most "difficult" case is:

$$\dfrac{\Gamma \vdash N \leq^{q} N' \rightrightarrows p\kappa_1 \to \kappa_2 \qquad p^{-1}\Gamma \vdash G \leq^{pq} G' \leftrightarrows \kappa_1}{\Gamma \vdash N\,G \leq^{q} N'\,G' \rightrightarrows \kappa_2} \qquad \dfrac{\Gamma \vdash N \leq^{q'} N' \rightrightarrows p'\kappa_1' \to \kappa_2' \qquad p'^{-1}\Gamma \vdash G \leq^{p'q'} G' \leftrightarrows \kappa_1'}{\Gamma \vdash N\,G \leq^{q'} N'\,G' \rightrightarrows \kappa_2'}$$

Let $q'' = \inf(q, q')$. By induction hypothesis 1, $\Gamma \vdash N \leq^{q''} N' \rightrightarrows p\kappa_1 \to \kappa_2$ and $p = p'$, $\kappa_1 = \kappa_1'$, and $\kappa_2 = \kappa_2'$. Hence, we can apply induction hypothesis 2 to obtain $p^{-1}\Gamma \vdash G \leq^{pq''} G' \leftrightarrows \kappa_1$ and conclude with rule AL-APP-NE. □

Transitivity is basically proven by induction on the sum of the lengths of the two given derivations. For the η-rules to go through we need to strengthen the induction hypothesis a bit; alternatively, one can use a different measure (Goguen, 2005).

Lemma 4.6 (Transitivity). Let q, q' such that $q'' = \sup(q, q') \neq \top$.

1 If $\mathcal{D}_1 :: \Gamma \vdash N_1 \leq^{q} N_2 \rightrightarrows \kappa$ and $\mathcal{D}_2 :: \Gamma \vdash N_2 \leq^{q'} N_3 \rightrightarrows \kappa'$ then $\kappa = \kappa'$ and $\Gamma \vdash N_1 \leq^{q''} N_3 \rightrightarrows \kappa$.
2 If $\mathcal{D}_1 :: \Gamma \vdash N_1 \leq^{q} N_2 \rightrightarrows \_$ and $\mathcal{D}_2 :: \Gamma \vdash N_2\,\vec{X} \leq^{q'} N \rightrightarrows \kappa$ then $\Gamma \vdash N_1\,\vec{X} \leq^{q''} N \rightrightarrows \kappa$.

3 If $\mathcal{D}_1 :: \Gamma \vdash N_1 \leq^q N_2 \rightrightarrows \_$ and $\mathcal{D}_2 :: \Gamma \vdash N_2\,\vec{X} \leq^{q'} W \leftleftarrows \kappa$ then $\Gamma \vdash N_1\,\vec{X} \leq^{q''} W \leftleftarrows \kappa$.

4 Symmetrical to 2: If $\mathcal{D}_1 :: \Gamma \vdash N \leq^q N_2\,\vec{X} \rightrightarrows \kappa$ and $\mathcal{D}_2 :: \Gamma \vdash N_2 \leq^{q'} N_3 \rightrightarrows \_$ then $\Gamma \vdash N \leq^{q''} N_3\,\vec{X} \rightrightarrows \kappa$.

5 Symmetrical to 3: If $\mathcal{D}_1 :: \Gamma \vdash W \leq^q N_2\,\vec{X} \leftleftarrows \kappa$ and $\mathcal{D}_2 :: \Gamma \vdash N_2 \leq^{q'} N_3 \rightrightarrows \_$ then $\Gamma \vdash W \leq^{q''} N_3\,\vec{X} \leftleftarrows \kappa$.

6 If $\mathcal{D}_1 :: \Gamma \vdash W_1 \leq^q W_2 \leftleftarrows \kappa$ and $\mathcal{D}_2 :: \Gamma \vdash W_2 \leq^{q'} W_3 \leftleftarrows \kappa$ then $\Gamma \vdash W_1 \leq^{q''} W_3 \leftleftarrows \kappa$.

A direct consequence of part 6 is the following corollary, which holds also for $q = \top$ or $q' = \top$.

Corollary 4.7 (Transitivity). If $\Gamma \vdash F_1 \leq^q F_2 \leftleftarrows \kappa$ and $\Gamma \vdash F_2 \leq^{q'} F_3 \leftleftarrows \kappa$ then $\Gamma \vdash F_1 \leq^{\sup(q,q')} F_3 \leftleftarrows \kappa$.

Proof of the transitivity lemma. Simultaneously by induction on $|\mathcal{D}_1| + |\mathcal{D}_2|$.

Case (part 1):

$$\textsc{al-app-ne}\ \frac{\Gamma \vdash N_1 \leq^q N_2 \rightrightarrows p\kappa_1 \to \kappa_2 \qquad p^{-1}\Gamma \vdash G_1 \leq^{pq} G_2 \leftleftarrows \kappa_1}{\Gamma \vdash N_1\,G_1 \leq^q N_2\,G_2 \rightrightarrows \kappa_2}$$

$$\textsc{al-app-ne}\ \frac{\Gamma \vdash N_2 \leq^{q'} N_3 \rightrightarrows p'\kappa_1' \to \kappa_2' \qquad p'^{-1}\Gamma \vdash G_2 \leq^{p'q'} G_3 \leftleftarrows \kappa_1'}{\Gamma \vdash N_2\,G_2 \leq^{q'} N_3\,G_3 \rightrightarrows \kappa_2'}$$

By induction hypothesis 1 we have $p = p'$, $\kappa_1 = \kappa_1'$ and $\kappa_2 = \kappa_2'$ and $\Gamma \vdash N_1 \leq^{q''} N_3 \rightrightarrows p\kappa_1 \to \kappa_2$. Since $\sup(pq, pq') = p\sup(q, q')$, by induction hypothesis 6 we get $\Gamma \vdash G_1 \leq^{pq''} G_3 : \kappa_1$, and we conclude by rule AL-APP-NE.

Case (part 2):

$$\Gamma \vdash N_1 \leq^q N_2 \rightrightarrows \_$$

$$\textsc{al-app-ne}\ \frac{\Gamma \vdash N_2\,\vec{X} \leq^{q'} N' \rightrightarrows p\kappa \to \kappa' \qquad p^{-1}\Gamma \vdash X \leq^{pq'} G' \leftleftarrows \kappa}{\Gamma \vdash N_2\,\vec{X}\,X \leq^{q'} N'\,G' \rightrightarrows \kappa'}$$

By induction hypothesis $\Gamma \vdash N_1\,\vec{X} \leq^{q''} N' \rightrightarrows p\kappa \to \kappa'$. By the weakening lemma, $p^{-1}\Gamma \vdash X \leq^{pq''} G' \leftleftarrows \kappa$. Hence, the claim follows by rule AL-APP-NE.

Case (part 3):

$$\Gamma \vdash N_1 \leq^q N_2 \rightrightarrows \_ \qquad \textsc{al-}\eta\textsc{-r}\ \frac{\Gamma, X : p\kappa \vdash N_2\,\vec{X}\,X \leq^{q'} F \leftleftarrows \kappa'}{\Gamma \vdash N_2\,\vec{X} \leq^{q'} \lambda X F \leftleftarrows p\kappa \to \kappa'}$$

By assumption, $F \searrow W$ and $\Gamma, X : p\kappa \vdash N_2\,\vec{X}\,X \leq^{q'} W \leftleftarrows \kappa'$. The induction hypothesis yields $\Gamma, X : p\kappa \vdash N_1\,\vec{X}\,X \leq^{q''} W \leftleftarrows \kappa'$, hence we can conclude with rule AL-η-R.

Parts 4 and 5 are proven analogously to parts 2 and 3. For part 6, if at least one inequality talks about two neutral constructors, we can use parts 3 and 5, respectively. The remaining cases are represented by:

Case Assume $X \notin \mathsf{FV}(N_1, N_3)$.

$$\textsc{al-}\eta\textsc{-r}\ \frac{\Gamma, X : p\kappa \vdash N_1\,X \leq^q F \leftleftarrows \kappa'}{\Gamma \vdash N_1 \leq^q \lambda X F \leftleftarrows p\kappa \to \kappa'} \qquad \textsc{al-}\eta\textsc{-l}\ \frac{\Gamma, X : p\kappa \vdash F \leq^q N_3\,X \leftleftarrows \kappa'}{\Gamma \vdash \lambda X F \leq^q N_3 \leftleftarrows p\kappa \to \kappa'}$$

By induction hypothesis, $\Gamma, X : p\kappa \vdash N_1\,X \leq^q N_3\,X \leftleftarrows \kappa'$. This derivation must have been

generated from $\Gamma, X:p\kappa \vdash N_1 \leq^q N_3 \rightrightarrows \kappa''$. By strengthening (Lemma 4.3), $\Gamma \vdash N_1 \leq^q N_3 \rightrightarrows \kappa''$. Applying AL-NE, we are done.

Case

$$\text{AL-}\eta\text{-R}\ \frac{\Gamma, X:p\kappa \vdash N\,X \leq^q F \leftleftarrows \kappa'}{\Gamma \vdash N \leq^q \lambda XF \leftleftarrows p\kappa \to \kappa'} \qquad \text{AL-}\lambda\ \frac{\Gamma, X:p\kappa \vdash F \leq^{q'} F' \leftleftarrows \kappa'}{\Gamma \vdash \lambda XF \leq^{q'} \lambda XF' \leftleftarrows p\kappa \to \kappa'}$$

By induction hypothesis and AL-η-R.

Case

$$\text{AL-}\lambda\ \frac{\Gamma, X:p\kappa \vdash F_1 \leq^q F_2 \leftleftarrows \kappa'}{\Gamma \vdash \lambda XF_1 \leq^q \lambda XF_2 \leftleftarrows p\kappa \to \kappa'} \qquad \text{AL-}\lambda\ \frac{\Gamma, X:p\kappa \vdash F_2 \leq^{q'} F_3 \leftleftarrows \kappa'}{\Gamma \vdash \lambda XF_2 \leq^{q'} \lambda XF_3 \leftleftarrows p\kappa \to \kappa'}$$

By induction hypothesis and AL-λ.

□

The next lemma states that the η-rules can be extended beyond neutral constructors. It can be proven directly:

Lemma 4.8 (Generalizing the η-rules).

1 If $\Gamma, Y:p\kappa \vdash F\,Y \leq^q F' \leftleftarrows \kappa'$ and $Y \notin \mathsf{FV}(F)$ then $\Gamma \vdash F \leq^q \lambda YF' \leftleftarrows p\kappa \to \kappa'$.
2 If $\Gamma, Y:p\kappa \vdash F' \leq^q F\,Y \leftleftarrows \kappa'$ and $Y \notin \mathsf{FV}(F)$ then $\Gamma \vdash \lambda YF' \leq^q F \leftleftarrows p\kappa \to \kappa'$.

Proof. We prove the first statement, the second is symmetrical. By assumption $F\,Y \searrow W$, and by Lemma 3.1 we can distinguish two cases:

Case $F \searrow N$ neutral. Then $\Gamma \vdash N \leq^q \lambda YF' \leftleftarrows p\kappa \to \kappa'$ by rule AL-η-R and the goal follows.
Case $F \searrow \lambda YG$ and $G \searrow W$. We get $\Gamma \vdash \lambda YG \leq^q \lambda YF' \leftleftarrows p\kappa \to \kappa'$ by rule AL-λ and again, the goal follows.

□

Now we come to the main lemma:

Lemma 4.9 (Substitution and application). Let $\Gamma \leq p\Delta$ and $\Delta \vdash G \leq^{pq} G' \leftleftarrows \kappa$.

1 If $\mathcal{D} :: \Gamma, X : p\kappa \vdash N \leq^q N' \rightrightarrows \kappa'$ then either $\Gamma \vdash [G/X]N \leq^q [G'/X]N' \rightrightarrows \kappa'$ or $\mathsf{rk}(\kappa') \leq \mathsf{rk}(\kappa)$ and $\Gamma \vdash [G/X]N \leq^q [G'/X]N' \leftleftarrows \kappa'$.
2 If $\mathcal{D} :: \Gamma, X : p\kappa \vdash W \leq^q W' \leftleftarrows \kappa'$ and $\Gamma \vdash W, W' : \kappa'$ then $\Gamma \vdash [G/X]W \leq^q [G'/X]W' \leftleftarrows \kappa'$.
3 If $\mathcal{D} :: \Gamma \vdash W \leq^q W' \leftleftarrows p\kappa \to \kappa'$ and $\Gamma \vdash W, W' : p\kappa \to \kappa'$ then $\Gamma \vdash W\,G \leq^q W'\,G' \leftleftarrows \kappa'$.

Note that part 1 works for ill-kinded neutral constructors as well, whereas parts 2 and 3 require well-kinded constructors. Well-kindedness is needed for the case AL-NE in part 2.

The three propositions are proven simultaneously by a lexicographic induction on $(\mathsf{rk}(\kappa), |\mathcal{D}|)$. It works because the constructor language is essentially the simply-typed λ-calculus (STL), which has a small proof-theoretical strength. The idea is taken from Joachimski and Matthes' proof of weak normalization for the STL (Joachimski and Matthes, 2003) which I have formalized in Twelf (Abel, 2004). The argument goes probably back to Anne Troelstra, it is implicit in Girard's combinatorial weak normalization proof (Girard et al., 1989, Ch. 4.3) and has been

reused by Watkins et. al. (2003) and Adams (2005, p. 65ff) to define a logical framework based solely on normal terms.

Before proving this lemma, we state some of its immediate consequences that can be proven directly (without induction), and, hence, we can use them on the induction hypotheses in the proof of the lemma.

Corollary 4.10 (Application for arbitrary constructors). Let $\Gamma \leq p\Delta$, $\Delta \vdash G_1 \leq^{pq} G_2 \leftleftarrows \kappa$, and $\Gamma \vdash F_1, F_2 : p\kappa \to \kappa'$. If $\Gamma \vdash F_1 \leq^{q} F_2 \leftleftarrows p\kappa \to \kappa'$ then $\Gamma \vdash F_1\, G_1 \leq^{q} F_2\, G_2 \leftleftarrows \kappa'$.

Proof. If $q = \top$, there is nothing to show. Otherwise, by assumption $F_1 \searrow W_1$, $F_2 \searrow W_2$, and $\Gamma \vdash W_1 \leq^{q} W_2 \leftleftarrows p\kappa \to \kappa'$. By subject reduction, $\Gamma \vdash W_1, W_2 : p\kappa \to \kappa'$, thus, we can apply the lemma. It yields $\Gamma \vdash W_1\, G_1 \leq^{q} W_2\, G_2 \leftleftarrows \kappa'$, meaning that $W_i\, G_i \searrow V_i$ and $\Gamma \vdash V_1 \leq^{q} V_2 \leftleftarrows \kappa'$. We are done, since by Lemma 3.1, $F_i\, G_i \searrow V_i$. □

Corollary 4.11 (Substitution with modified contexts). Let $\Gamma \leq p\Delta$, $\Delta \vdash G \leq^{pq} G' \leftleftarrows \kappa$, and $p_0^{-1}(\Gamma, X : p\kappa) \vdash F, F' : \kappa'$. If $p_0^{-1}(\Gamma, X : p\kappa) \vdash F \leq^{p_0 q} F' \leftleftarrows \kappa'$ then $p_0^{-1}\Gamma \vdash [G/X]F \leq^{p_0 q} [G'/X]F' \leftleftarrows \kappa'$.

Proof. By assumption, $F \searrow W$, $F' \searrow W'$, and $p_0^{-1}(\Gamma, X : p\kappa) \vdash W \leq^{p_0 q} W' \leftleftarrows \kappa'$. Subject reduction entails $p_0^{-1}(\Gamma, X : p\kappa) \vdash W, W' : \kappa'$. We need to show

$$p_0^{-1}\Gamma \vdash [G/X]W \leq^{p_0 q} [G'/X]W' \leftleftarrows \kappa',$$

then the corollary follows by properties of weak head evaluation (Lemma 3.1).

We distinguish cases on p and p_0. If $p_0 q = \top$, there is nothing to show, thus, in the following, we can exclude $q = \top$ and $p_0 = \top$.

Case $p_0 = +$. The goal is a direct instance of the lemma.

Case $p_0^{-1} p = \top$. Setting $p' := \top$, $q' := p_0 q$, and $\Gamma' := p_0^{-1}\Gamma$, the assumption simplifies to $\Gamma', X : p'\kappa \vdash W \leq^{q'} W \leftleftarrows \kappa'$. We have $\Gamma' \leq p'\Delta$ (by taking the inverse of p_0 on $\Gamma \leq p\Delta$), and trivially $\Delta \vdash G \leq^{p'q'} G' \leftleftarrows \kappa$, since $p'q' = \top$. Now the goal is an instance of the lemma with Γ', p', and q'.

Case $p_0 = \circ$. If $p \neq \circ$, then $p_0^{-1} p = \top$, and this case has already been treated. Otherwise, $p = \circ$ and our assumptions simplify to $\Delta \vdash G \leq^{\circ\circ} G' \leftleftarrows \kappa$ and $\circ^{-1}\Gamma,\ X : \circ\kappa \vdash W \leq^{\circ} W' \leftleftarrows \kappa'$. The goal $\circ^{-1}\Gamma \vdash [G/X]W \leq^{\circ} [G'/X]W' : \kappa'$ follows now from the lemma, since from the assumption $\Gamma \leq \circ\Delta = \circ\circ\Delta$ we obtain $\circ^{-1}\Gamma \leq \circ\Delta$.

Case $p_0 = -$. The assumptions can be written as $\Delta \vdash G \leq^{(-p)(-q)} G' : \kappa$ and $-\Gamma, X : (-p)\kappa \vdash W \leq^{-q} W' \leftleftarrows \kappa'$. Since $\Gamma \leq p\Delta$ entails $-\Gamma \leq (-p)\Delta$, our goal follows from the lemma.

□

Proof of Lemma 4.9. The three statements are shown simultaneously by lexicographic induction on $(\mathsf{rk}(\kappa), |\mathcal{D}|)$. First, we prove the substitution lemma for neutral constructors:

Case $p \leq +$ and $\kappa' = \kappa$ and

$$\mathcal{D} = \frac{}{\Gamma,\ X : p\kappa \vdash X \leq^{q} X \rightrightarrows \kappa}$$

Since $pq \leq q$ and $\Gamma \leq p\Delta \leq \Delta$, we can weaken the assumption $\Delta \vdash G \leq^{pq} G' \leftleftarrows \kappa$ to $\Gamma \vdash G \leq^{q} G' \leftleftarrows \kappa$ and have trivially $\mathsf{rk}(\kappa') \leq \mathsf{rk}(\kappa)$.

Case

$$\mathcal{D} = \frac{(Y : p'\kappa') \in \Gamma \qquad p' \leq +}{\Gamma,\, X : p\kappa \vdash Y \leq^q Y \rightrightarrows \kappa'}$$

Then of course $\Gamma \vdash Y \leq^q Y \rightrightarrows \kappa'$.

Case

$$\mathcal{D} = \frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ \Gamma, X : p\kappa \vdash N \leq^q N' \rightrightarrows p_0\kappa_0 \to \kappa' & p_0^{-1}(\Gamma, X : p\kappa) \vdash I \leq^{p_0 q} I' \leftrightarrows \kappa_0\end{array}}{\Gamma, X : p\kappa \vdash N\, I \leq^q N'\, I' \rightrightarrows \kappa'}$$

Let $M = [G/X]N$, $M' = [G'/X]N'$, $J = [G/X]I$, and $J' = [G'/X]I'$. Using Cor. 4.11, the induction hypothesis 2 on $\mathcal{D}_2$ yields $p_0^{-1}\Gamma \vdash J \leq^{p_0 q} J' \leftrightarrows \kappa_0$. Induction hypothesis 1 on $\mathcal{D}_1$ has two possible results:

Subcase $\Gamma \vdash M \leq^q M' \rightrightarrows p_0\kappa_0 \to \kappa'$. Apply AL-APP-NE.

Subcase $\Gamma \vdash M \leq^q M' \leftrightarrows p_0\kappa_0 \to \kappa'$. Since $\mathsf{rk}(\kappa_0) < \mathsf{rk}(p_0\kappa_0 \to \kappa') \leq \mathsf{rk}(\kappa)$, we can use induction hypothesis 3 to obtain our goal $\Gamma \vdash M\, J \leq^q M'\, J' \leftrightarrows \kappa'$. The required inequation $\mathsf{rk}(\kappa') \leq \mathsf{rk}(\kappa)$ follows trivially.

Secondly, we consider some cases of substitution into checking-mode subtyping derivations:

Case

$$\mathcal{D} = \frac{\Gamma, X : p\kappa \vdash N \leq^q N' \rightrightarrows \kappa''}{\Gamma, X : p\kappa \vdash N \leq^q N' \leftrightarrows \kappa'}$$

By induction hypothesis, either $\Gamma \vdash [G/X]N \leq^q [G'/X]N' \rightrightarrows \kappa''$. Then we simply apply rule AL-NE. Or $\Gamma \vdash [G/X]N \leq^q [G'/X]N' \leftrightarrows \kappa''$. Since $\Gamma, X : p\kappa \vdash N, N' : \kappa'$, we have by the soundness theorem (Thm 3.4) that $\kappa'' \leq \kappa'$. By the weakening lemma, $\Gamma \vdash [G/X]N \leq^q [G'/X]N' \leftrightarrows \kappa'$.

Case $\kappa' = p_1\kappa_1 \to \kappa_2$ and

$$\mathcal{D} = \frac{\Gamma, Y : p_1\kappa_1, X : p\kappa \vdash N\, Y \leq^q F \leftrightarrows \kappa_2}{\Gamma, X : p\kappa \vdash N \leq^q \lambda Y F \leftrightarrows p_1\kappa_1 \to \kappa_2}.$$

W. l. o. g., $Y \notin \mathsf{FV}(G, G')$. Since $\Gamma, X : p\kappa \vdash N, \lambda Y F : p_1\kappa_1 \to \kappa_2$ by assumption, we get $\Gamma, Y : p_1\kappa_1, X : p\kappa \vdash N\, Y, F : \kappa_2$ by inversion and weakening on kinding. Hence, we can apply the induction hypothesis yielding $\Gamma, Y : p_1\kappa_1 \vdash ([G/X]N)\, Y \leq^q [G'/X]F \leftrightarrows \kappa_2$. Thus, $\Gamma \vdash [G/X]N \leq^q [G'/X](\lambda Y F) \leftrightarrows p_1\kappa_1 \to \kappa_2$ by the generalized η-rule (Lemma 4.8).

Finally, we turn our attention to the application lemma:

Case

$$\mathcal{D} = \frac{\Gamma \vdash N \leq^q N' \rightrightarrows \kappa'}{\Gamma \vdash N \leq^q N' \leftrightarrows p\kappa_1 \to \kappa_2}$$

By the soundness theorem, $\kappa' = p'\kappa_1' \to \kappa_2' \leq p\kappa_1 \to \kappa_2$. We weaken the assumption to $\Delta \vdash G \leq^{p' q} G' \leftrightarrows \kappa_1'$, and conclude by AL-APP-NE and AL-NE.

Case

$$\mathcal{D} = \frac{\Gamma, X : p\kappa \vdash N\, X \leq^q F \leftrightarrows \kappa'}{\Gamma \vdash N \leq^q \lambda X F \leftrightarrows p\kappa \to \kappa'}\; X \notin \mathsf{FV}(N)$$

By induction hypothesis 2 (with Cor. 4.11) we get $\Gamma \vdash N\, G \leq^q [G/X]F \leftrightarrows \kappa'$. We are done, since $(\lambda X F)\, G$ and $[G/X]F$ have the same weak head normal form.

□

Theorem 4.12 (Completeness).

1 If $\mathcal{D} :: \Gamma \vdash F : \kappa$ then $\Gamma \vdash F \leq^q F \leftrightarrows \kappa$.
2 If $\mathcal{D} :: \Gamma \vdash F = F' : \kappa$ then $\Gamma \vdash F \leq^\circ F' \leftrightarrows \kappa$.
3 If $\mathcal{D} :: \Gamma \vdash F \leq F' : \kappa$ then $\Gamma \vdash F \leq^+ F' \leftrightarrows \kappa$.

Proof. Simultaneously by induction on $\mathcal{D}$.

Case KIND-C, EQ-C: Use AL-C and AL-NE.

Case KIND-VAR, EQ-VAR: Use AL-VAR and AL-NE.

Case KIND-λ, EQ-λ, LEQ-λ: Use AL-λ on the induction hypothesis.

Case KIND-APP, EQ-APP: Use application lemma (Cor. 4.10) on the induction hypotheses.

Case

$$\text{EQ-}\beta\ \frac{\Gamma, X : p\kappa \vdash F : \kappa' \qquad p^{-1}\Gamma \vdash G : \kappa}{\Gamma \vdash (\lambda X F)\, G = [G/X]F : \kappa'}$$

By induction hypothesis, $\Gamma, X : p\kappa \vdash F \leq^\circ F \leftrightarrows \kappa'$ and $p^{-1}\Gamma \vdash G \leq^{p\circ} G' \leftrightarrows \kappa$. By the substitution lemma (Cor. 4.11) we get $\Gamma \vdash [G/X]F \leq^\circ [G/X]F \leftrightarrows \kappa$. Since $[G/X]F \searrow W$ implies $(\lambda X F)\, G \searrow W$, we are done.

Case

$$\text{EQ-}\eta\ \frac{\Gamma \vdash F : p\kappa \to \kappa'}{\Gamma \vdash (\lambda X.\, F\, X) = F : p\kappa \to \kappa'}$$

By induction hypothesis, $\Gamma \vdash F \leq^\circ F \leftrightarrows p\kappa \to \kappa'$, and we can use the application lemma to get $\Gamma, X : p\kappa \vdash F\, X \leq^\circ F\, X \leftrightarrows \kappa'$. Using the generalized η-rule (Lemma 4.8), we infer $\Gamma \vdash \lambda X.\, F\, X \leq^\circ F \leftrightarrows p\kappa \to \kappa'$.

Case

$$\text{EQ-}\top\ \frac{\Gamma \vdash F : \top\kappa \to \kappa' \qquad \top^{-1}\Gamma \vdash G : \kappa \qquad \top^{-1}\Gamma \vdash G' : \kappa}{\Gamma \vdash F\, G = F\, G' : \kappa'}$$

By induction hypothesis $\Gamma \vdash F \leq^\circ F \leftrightarrows \top\kappa \to \kappa'$. Since trivially $\top^{-1}\Gamma \vdash G \leq^\top G' \leftrightarrows \kappa$, the goal follows by the application lemma.

Case EQ-SYM: Use swapping lemma (4.4).

Case EQ-TRANS, LEQ-TRANS: Use transitivity lemma.

Case LEQ-REFL: Use weakening lemma.

Case LEQ-ANTISYM:

$$\text{LEQ-ANTISYM}\ \frac{\Gamma \vdash F \leq F' : \kappa \qquad \Gamma \vdash F' \leq F : \kappa}{\Gamma \vdash F = F' : \kappa}$$

By induction hypothesis $\Gamma \vdash F \leq^+ F' \leftrightarrows \kappa$ and $\Gamma \vdash F' \leq^+ F \leftrightarrows \kappa$. Swapping the second judgement with Lemma 4.4, we can apply the antisymmetry lemma to infer $\Gamma \vdash F' \leq^\circ F \leftrightarrows \kappa$, since $\inf(+, -) = \circ$.

Case LEQ-FUN, LEQ-ARG$+$, LEQ-ARG$-$: Use application lemma (Cor. 4.10), in the last case also swapping.

□

Now we have a sound and complete subtyping algorithm, but we have nothing yet to get it started. Since there are no subtyping assumptions or basic subtyping relations (like $\mathsf{Nat} \leq \mathsf{Real}$),

two related constructors are already equal. In the next section we will extend the subtyping relation to make it more meaningful.

5. Extension to Sized Types

We extend our type language by size expressions a, which have kind ord, and sized-type constructors of kind $p\mathsf{ord} \to \kappa$. This leads to the following grammar for kinds.

$$\begin{array}{lcl} \hat{\kappa} & ::= & \kappa \mid \mathsf{ord} \\ \kappa & ::= & * \mid p\hat{\kappa} \to \kappa \end{array}$$

Note that there are no functions into ord, i. e., no kinds $p\hat{\kappa} \to \mathsf{ord}$. Size expressions obey a simple grammar, they can only be successors of zero or a of variable, or be infinity. (The successor of infinity is identified with infinity, see below.)

$$\begin{array}{llcll} \mathsf{SExp} & \ni & a, b & ::= & X \mid \mathsf{0} \mid \mathsf{s}\,a \mid \infty & \text{size expressions} \\ \mathsf{Constr} & \ni & A, B, F, G, H & ::= & a \mid C \mid X \mid \lambda X F \mid F\,G & \text{type constructors} \end{array}$$

The constants C are drawn from an extended signature Σ, which features sum, product, inductive, and coinductive types:

$$\begin{array}{lcll} 1 & : & * & \text{unit type} \\ + & : & * \xrightarrow{+} * \xrightarrow{+} * & \text{disjoint sum} \\ \times & : & * \xrightarrow{+} * \xrightarrow{+} * & \text{cartesian product} \\ \mu_{\kappa_*} & : & \mathsf{ord} \xrightarrow{+} (\kappa_* \xrightarrow{+} \kappa_*) \xrightarrow{+} \kappa_* & \text{inductive constructors} \\ \nu_{\kappa_*} & : & \mathsf{ord} \xrightarrow{-} (\kappa_* \xrightarrow{+} \kappa_*) \xrightarrow{+} \kappa_* & \text{coinductive constructors} \end{array}$$

The first argument to μ_κ and ν_κ shall be written as superscript. Inductive types $\mu^a_{\kappa_*} F$ and coinductive types $\nu^a_{\kappa_*} F$ can only be of kinds that do not mention ord. These are called *pure kinds* from here and denoted by κ_*.

In the extended signature we can, for instance, model lists of length $< n$ as $\mathsf{List}^a A := \mu^a_* \lambda X.\, 1 + A \times X$ where $a = \mathsf{s}(\mathsf{s} \ldots (\mathsf{s}\,\mathsf{0}))$ (n times s). Streams of depth of at least n are represented as $\mathsf{Stream}^a\, A := \nu^a_* \lambda X.\, A \times X$. The type $\mathsf{List}^\infty A$ contains lists or arbitrary length, and $\mathsf{Stream}^\infty A$ productive streams (which never run out of elements). We call such types with an ordinal index *sized*. Naturally, lists are covariant in their size argument and streams are contravariant (each stream which produces at least $n+1$ elements produces of course also at least n elements).

Barthe et. al. (2004) define a calculus $\lambda\hat{}$ with sized inductive and coinductive types in which all recursive functions are terminating and all streams are productive. Their sizes follow the same grammar, but are called *stage expressions*. Sized data types are introduced by a set of data constructors—we can define them using μ and ν. The normalization property is ensured by a restricted typing rule for recursion; in our notation it reads

$$\frac{\imath : \mathsf{ord}, f : \mu^\imath_* F \to G\,\imath \vdash e : \mu^{\mathsf{s}\,\imath}_* F \to G\,(\mathsf{s}\,\imath) \qquad F : * \xrightarrow{+} * \qquad G : \mathsf{ord} \xrightarrow{+} *}{(\mathsf{letrec}\ f = e) : \forall_{\mathsf{ord}} \lambda\imath.\, \mu^\imath_* F \to G\,\imath}.$$

Similarly, corecursive functions are introduced by

$$\frac{\imath : \mathsf{ord}, f : \vec{G}\,\imath \to \nu^{\imath}_{*} F \vdash e : \vec{G}\,(\mathsf{s}\,\imath) \to \nu^{\mathsf{s}\,\imath}_{*} F \qquad F : * \xrightarrow{+} * \qquad \vec{G} : \mathsf{ord} \xrightarrow{-} *}{(^{\mathsf{co}}\mathsf{letrec}\ f = e) : \forall_{\mathsf{ord}} \lambda\imath.\, \vec{G}\,\imath \to \nu^{\imath}_{*} F}.$$

We have reduced the stage expressions of $\lambda\widehat{\ }$ to just constructors of a special kind and model the inclusion between sized types of different stages simply by variance. Lifting the restriction of $\lambda\widehat{\ }$ that type constructors must be monotone in all arguments comes at no cost in our formulation: We can define the type of A-labeled, B-branching trees as $\mathsf{Tree}^{a}\, A\, B = \mu^{a}_{*} \lambda X.\, 1 + A \times (B \to X)$, where now

$$\mathsf{Tree} : \mathsf{ord} \xrightarrow{+} * \xrightarrow{+} * \xrightarrow{-} *.$$

Higher-order subtyping becomes really necessary when we allow inductive *constructors* instead of inductive *types*. These are necessary to model higher-order and heterogeneous (also called nested) datatypes, as for instance powerlists:

$$\begin{array}{lcl} \mathsf{PList} & : & \mathsf{ord} \xrightarrow{+} * \xrightarrow{+} * \\ \mathsf{PList}^{a} & := & \mu^{a}_{+*\to *} \lambda X \lambda A.\, A + X\,(A \times A) \end{array}$$

Many more examples for datatypes of this kind can be found in the literature (Okasaki, 1999; Altenkirch and Reus, 1999; Bird and Paterson, 1999; Hinze, 2000; Abel et al., 2005).

Having the machinery of higher-order subtyping running, the extensions needed for sized types are minimal:

Kinding, equality and subtyping. These judgements are now considered w. r. t. the extended kind and constructor grammar. Additional rules are:

$$\textsc{kind-0}\ \frac{}{\Gamma \vdash 0 : \mathsf{ord}} \qquad \textsc{kind-s}\ \frac{\Gamma \vdash a : \mathsf{ord}}{\Gamma \vdash \mathsf{s}\,a : \mathsf{ord}} \qquad \textsc{kind-}\infty\ \frac{}{\Gamma \vdash \infty : \mathsf{ord}}$$

$$\textsc{eq-s-}\infty\ \frac{}{\Gamma \vdash \mathsf{s}\,\infty = \infty : \mathsf{ord}}$$

$$\textsc{eq-0}\ \frac{}{\Gamma \vdash 0 = 0 : \mathsf{ord}} \qquad \textsc{eq-s}\ \frac{\Gamma \vdash a = a' : \mathsf{ord}}{\Gamma \vdash \mathsf{s}\,a = \mathsf{s}\,a' : \mathsf{ord}} \qquad \textsc{eq-}\infty\ \frac{}{\Gamma \vdash \infty = \infty : \mathsf{ord}}$$

$$\textsc{leq-0}\ \frac{\Gamma \vdash a : \mathsf{ord}}{\Gamma \vdash 0 \leq a : \mathsf{ord}} \qquad \textsc{leq-s-r}\ \frac{\Gamma \vdash a : \mathsf{ord}}{\Gamma \vdash a \leq \mathsf{s}\,a : \mathsf{ord}} \qquad \textsc{leq-}\infty\ \frac{\Gamma \vdash a : \mathsf{ord}}{\Gamma \vdash a \leq \infty : \mathsf{ord}}$$

$$\textsc{leq-s}\ \frac{\Gamma \vdash a \leq a' : \mathsf{ord}}{\Gamma \vdash \mathsf{s}\,a \leq \mathsf{s}\,a' : \mathsf{ord}}$$

Weak head normal forms and evaluation contexts.

$$\begin{array}{llcll} \mathsf{Ne} & \ni & N & ::= & C \mid X \mid N\,G \qquad \text{neutral constructors} \\ \mathsf{Ne}_{\mathsf{ord}} & \ni & n & ::= & X \mid 0 \mid \mathsf{s}\,n \qquad \text{neutral size expressions} \\ \mathsf{Val}_{\mathsf{ord}} & \ni & w & ::= & n \mid \infty \qquad \text{size values} \\ \mathsf{Val} & \ni & W & ::= & w \mid N \mid \lambda X F \qquad \text{weak head values} \end{array}$$

Weak head evaluation is extended by the following rules:

$$\text{EVAL-S-}\infty\ \frac{a \searrow \infty}{\mathsf{s}\,a \searrow \infty} \qquad \text{EVAL-S}\ \frac{a \searrow n}{\mathsf{s}\,a \searrow \mathsf{s}\,n} \qquad \text{EVAL-0}\ \frac{}{0 \searrow 0} \qquad \text{EVAL-}\infty\ \frac{}{\infty \searrow \infty}$$

Weak head evaluation is still deterministic.

Lemma 5.1 (Soundness of weak head evaluation and completeness of size evaluation).

1 If $\Gamma \vdash F : \hat{\kappa}$ and $F \searrow W$ then $\Gamma \vdash F = W : \hat{\kappa}$.
2 If $\Gamma \vdash a : \mathsf{ord}$ then $a \searrow w$ for some w.
3 If $\Gamma \vdash a = a' : \mathsf{ord}$ then $a \searrow w$ and $a' \searrow w$ for some w.

Neutral size values are compared w. r. t. the following rules:

$$\text{COMP-VAR}\ \frac{}{X \leq X} \qquad \text{COMP-S-R}\ \frac{X \leq n'}{X \leq \mathsf{s}\,n'} \qquad \text{COMP-0}\ \frac{}{0 \leq n} \qquad \text{COMP-S}\ \frac{n \leq n'}{\mathsf{s}\,n \leq \mathsf{s}\,n'}$$

Neutral size value comparison is reflexive, transitive and antisymmetric.

Lemma 5.2 (Soundness and completeness of size comparison).

1 If $\Gamma \vdash n, n' : \mathsf{ord}$ and $n \leq n'$ then $\Gamma \vdash n \leq n' : \mathsf{ord}$.
2 If $\Gamma \vdash a \leq a' : \mathsf{ord}$ then $a \searrow w$ and $a' \searrow w'$ and either $w' = \infty$ or $w \leq w'$.

Algorithmic subtyping. Now we can extend algorithmic subtyping to kinds $\hat{\kappa}$. We need to cover the case $\hat{\kappa} = \mathsf{ord}$:

$$\begin{array}{lcl}
\Gamma \vdash a \leq^{\top} a' \leftleftarrows \mathsf{ord} & \iff & \text{true} \\
\Gamma \vdash a \leq^{\circ} a' \leftleftarrows \mathsf{ord} & \iff & a \searrow w \text{ and } a' \searrow w \\
\Gamma \vdash a \leq^{+} a' \leftleftarrows \mathsf{ord} & \iff & a \searrow w \text{ and } a' \searrow w' \text{ and } w' = \infty \text{ or } w \leq w' \\
\Gamma \vdash a \leq^{-} a' \leftleftarrows \mathsf{ord} & \iff & a \searrow w \text{ and } a' \searrow w' \text{ and } w = \infty \text{ or } w' \leq w
\end{array}$$

Lemma 5.3 (Soundness of algorithmic subtyping for ord). Let $\Gamma \vdash a, a' : \mathsf{ord}$.

1 If $\Gamma \vdash a \leq^{\circ} a' \leftleftarrows \mathsf{ord}$ then $\Gamma \vdash a = a' : \mathsf{ord}$.
2 If $\Gamma \vdash a \leq^{+} a' \leftleftarrows \mathsf{ord}$ then $\Gamma \vdash a \leq a : \mathsf{ord}$.
3 If $\Gamma \vdash a \leq^{-} a' \leftleftarrows \mathsf{ord}$ then $\Gamma \vdash a' \leq a : \mathsf{ord}$

Proof. For part 1, use soundness of weak head evaluation. Part 2 uses LEQ-∞ in case $a' \searrow \infty$, and soundness of size comparison otherwise. Part 3 analogously. □

Lemma 5.4 (Completeness of algorithmic subtyping for ord).

1 If $\Gamma \vdash a : \mathsf{ord}$ then $\Gamma \vdash a \leq^{q} a \leftleftarrows \mathsf{ord}$.
2 If $\Gamma \vdash a = a' : \mathsf{ord}$ then $\Gamma \vdash a \leq^{\circ} a' \leftleftarrows \mathsf{ord}$.
3 If $\Gamma \vdash a \leq a' : \mathsf{ord}$ then $\Gamma \vdash a \leq^{+} a' \leftleftarrows \mathsf{ord}$.

Proof. For part 1, we have $a \searrow w$ by Lemma 5.1. In case $q = \circ$ or $w = \infty$, we are done, for $q \in \{+, -\}$ use reflexivity of size comparison for neutral w. Part 2 is an instance of Lemma 5.1, and part 3 of Lemma 5.2. □

Theorem 5.5. Algorithmic subtyping for the extended system is still sound and complete.

Proof. In case $\hat{\kappa} = \mathsf{ord}$ use the previous lemmata, otherwise, we can replay the soundness and completeness proof of the previous chapters almost literally. □

6. Conclusion and Related Work

We have presented algorithmic subtyping for polarized F^ω, without bounded quantification, but with rules for η-equality and subkinding. The algorithm is economic since it computes the β-normal form incrementally, at each step just enough to continue with the subtyping test. Its completeness proof is quite short compared to completeness proofs of related systems in the literature (Steffen, 1998; Compagnoni and Goguen, 2003). However, it is unclear whether the proof scales to bounded quantification—this is worthwhile investigating in the future. From the perspective of proof theory, it should work, but the technical details have to be sorted out.

Related work. The inspiration for the algorithmic subtyping judgement presented here came from Coquand's $\beta\eta$-conversion algorithm (Coquand, 1991) and the idea for the crucial substitution and application lemma (4.9) from Joachimski and Matthes' proof of weak normalization for the simply-typed λ-calculus (Joachimski and Matthes, 2003). Both Coquand's algorithm and Joachimski and Matthes' characterization of weakly normalizing terms, originally due to van Raamsdonk et. al. (1999), bear strong resemblances to Goguen's typed operational semantics (Goguen, 1995, 1999).

Our algorithmic subtyping is closely related to Compagnoni and Goguen's weak-head subtyping (1999; 2003; 2006), but they are additionally dealing with bounded quantification and require neutral constructors to be fully β-normalized. They do not, however, treat η-equality and polarities.

Pierce and Steffen (1997) show decidability of higher-order subtyping with bounded quantification. Their calculus of strong cut-free subtyping is similar to our subtyping algorithm, only that they fully β-normalize the compared constructors and do not treat η. Steffen (1998) extends this work to polarities; in his first formulation, kinding and subtyping are mutually dependent. He resolves this issue by introducing a judgement for variable occurrence. Matthes and I (Abel and Matthes, 2004) have independently of Steffen developed a polarized version of F^ω which unifies variable occurrence and kinding through polarized contexts. Duggan and Compagnoni (1999) investigate subtyping for polarized object type constructors. Their system is similar to Steffen's, albeit without constructor-level λ-abstraction, hence there is no need to care for η.

Watkins et. al. (2003) and Adams (2005) construct logical frameworks solely based on normal terms. This could also be carried out for System $\mathsf{F}^\omega_{\leq}$. Application of two normal terms is immediately normalized, requiring also normalization of substitution. Termination of application and substitution could then also be proven by a lexicographic induction on kind and term. A subtyping algorithm for normal terms can be defined analogously to the one in this article, only that weak head evaluation could be omitted.

Acknowledgments. Thanks to Thierry Coquand and Ralph Matthes for inspiring discussions and to my former colleagues at Chalmers, especially the bosses of my project, for giving me time to work on my PhD thesis. Thanks to Healfdene Goguen and the anonymous referees for reading draft versions of this article and giving helpful and encouraging comments.

References

Abel, A. (2004). Weak normalization for the simply-typed lambda-calculus in Twelf, *Logical Frameworks and Metalanguages (LFM 04)*, IJCAR, Cork, Ireland.

Abel, A. and Matthes, R. (2004). Fixed points of type constructors and primitive recursion, *in* J. Marcinkowski and A. Tarlecki (eds), *Computer Science Logic, 18th Int. Workshop, CSL 2004, 13th Annual Conf. of the EACSL*, Vol. 3210 of *Lect. Notes in Comput. Sci.*, Springer-Verlag, pp. 190–204.

Abel, A., Matthes, R. and Uustalu, T. (2005). Iteration schemes for higher-order and nested datatypes, *Theoretical Computer Science* **333**(1–2): 3–66.

Adams, R. (2005). *A Modular Hierarchy of Logical Frameworks*, PhD thesis, University of Manchester.

Altenkirch, T. and Reus, B. (1999). Monadic presentations of lambda terms using generalized inductive types, *in* J. Flum and M. Rodríguez-Artalejo (eds), *Computer Science Logic, 13th Int. Workshop, CSL '99, 8th Annual Conf. of the EACSL*, Vol. 1683 of *Lect. Notes in Comput. Sci.*, Springer-Verlag, pp. 453–468.

Barthe, G., Frade, M. J., Giménez, E., Pinto, L. and Uustalu, T. (2004). Type-based termination of recursive definitions, *Math. Struct. in Comput. Sci.* **14**(1): 1–45.

Bird, R. S. and Paterson, R. (1999). De Bruijn notation as a nested datatype, *J. Func. Program.* **9**(1): 77–91.

Compagnoni, A. B. and Goguen, H. (1999). Anti-symmetry of higher-order subtyping, *in* J. Flum and M. Rodríguez-Artalejo (eds), *Computer Science Logic, 13th Int. Workshop, CSL '99, 8th Annual Conf. of the EACSL*, Vol. 1683 of *Lect. Notes in Comput. Sci.*, Springer-Verlag, pp. 420–438.

Compagnoni, A. B. and Goguen, H. (2003). Typed operational semantics for higher-order subtyping, *Inf. Comput.* **184**(2): 242–297.

Compagnoni, A. and Goguen, H. (2006). Anti-symmetry of higher-order subtyping and equality by subtyping, *Math. Struct. in Comput. Sci.* **16**: 41–65.

Coquand, T. (1991). An algorithm for testing conversion in type theory, *in* G. Huet and G. Plotkin (eds), *Logical Frameworks*, Cambridge University Press, pp. 255–279.

Davies, R. and Pfenning, F. (2000). Intersection types and computational effects, *Proc. of the 5th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP 2000)*, pp. 198–208.

Duggan, D. and Compagnoni, A. (1999). Subtyping for object type constructors. Presented at FOOL 6.

Girard, J.-Y., Lafont, Y. and Taylor, P. (1989). *Proofs and Types*, Vol. 7 of *Cambridge Tracts in Theoretical Computer Science*, Cambridge University Press.

Goguen, H. (1995). Typed operational semantics, *in* M. Deziani-Ciancaglini and G. D. Plotkin (eds), *Proc. of the 2nd Int. Conf. on Typed Lambda Calculi and Applications, TLCA '95*, Vol. 902 of *Lect. Notes in Comput. Sci.*, Springer-Verlag, pp. 186–200.

Goguen, H. (1999). Soundness of the logical framework for its typed operational semantics, *in* J.-Y. Girard (ed.), *Proc. of the 4th Int. Conf. on Typed Lambda Calculi and Applications, TLCA'99*, Vol. 1581 of *Lect. Notes in Comput. Sci.*, Springer-Verlag.

Goguen, H. (2005). Justifying algorithms for $\beta\eta$ conversion, *in* V. Sassone (ed.), *Foundations of Software Science and Computational Structures, FOSSACS 2005, Edinburgh, UK*, Vol. 3441 of *Lect. Notes in Comput. Sci.*, Springer-Verlag, pp. 410–424.

Harper, R. and Pfenning, F. (2005). On equivalence and canonical forms in the LF type theory, *ACM Transactions on Computational Logic* **6**(1): 61–101.

Hinze, R. (2000). Generalizing generalized tries, *J. Func. Program.* **10**(4): 327–351.

Hughes, J., Pareto, L. and Sabry, A. (1996). Proving the correctness of reactive systems using sized types, *Proc. of the 23rd ACM Symp. on Principles of Programming Languages, POPL'96*, pp. 410–423.

Joachimski, F. and Matthes, R. (2003). Short proofs of normalization, *Archive of Mathematical Logic* **42**(1): 59–87.

Mendler, N. P. (1987). Recursive types and type constraints in second-order lambda calculus, *Proc. of the 2nd IEEE Symp. on Logic in Computer Science (LICS'87)*, IEEE Computer Soc. Press, pp. 30–36.

Okasaki, C. (1999). From fast exponentiation to square matrices: An adventure in types, *Proc. of the 4th ACM SIGPLAN Int. Conf. on Functional Programming (ICFP '99)*, pp. 28–35.

Pierce, B. C. (2002). *Types and Programming Languages*, MIT Press.

Pierce, B. C. and Steffen, M. (1997). Higher order subtyping, *Theor. Comput. Sci.* **176**(1,2): 235–282.

Steffen, M. (1998). *Polarized Higher-Order Subtyping*, PhD thesis, Technische Fakultät, Universität Erlangen.

van Raamsdonk, F., Severi, P., Sørensen, M. H. and Xi, H. (1999). Perpetual reductions in lambda calculus, *Inf. Comput.* **149**(2): 173–225.

Watkins, K., Cervesato, I., Pfenning, F. and Walker, D. (2003). A concurrent logical framework I: Judgements and properties, *Technical report*, School of Computer Science, Carnegie Mellon University, Pittsburgh.