

Higher-Order Subtyping And Its Decidability

Adriana Compagnoni

abc@cs.stevens-tech.edu

Department of Computer Science
Stevens Institute of Technology
Castel Point on Hudson
Hoboken, NJ, 07030, USA

Abstract

We define the typed lambda calculus F_{\wedge}^{ω} (F-omega-meet), a natural generalization of Girard's system F^{ω} (F-omega) with intersection types and bounded polymorphism. A novel aspect of our presentation is the use of term rewriting techniques to present intersection types, which clearly splits the computational semantics (reduction rules) from the syntax (inference rules) of the system. We establish properties such as Church-Rosser for the reduction relation on types and terms, and Strong Normalization for the reduction on types. We prove that types are preserved by computation (Subject Reduction), and that the system satisfies the Minimal Types property. We define algorithms for type checking and subtype checking.

The development culminates with the proof of decidability of typing in F_{\wedge}^{ω} , containing the first proof of decidability of subtyping of a higher-order lambda calculus with subtyping.

Keywords: Higher-Order Lambda Calculus, Higher-Order Subtyping, Intersection Types, Bounded Polymorphism, Decidability.

Contents

1	Introduction	2
1.1	Motivation	2
1.2	Metatheory of F_{\wedge}^{ω}	3
1.3	Alternative presentations of the subtyping relation	3
1.4	Decidability of F_{\wedge}^{ω}	3
1.5	Other decidability proofs	4
1.6	Results	5
2	Syntax of F_{\wedge}^{ω}	6
2.1	Syntactic categories	6
2.2	Contexts	7
2.3	Context formation	8
2.4	Type formation	8
2.5	Subtyping	9
2.6	Term formation	9
2.7	Discussion	10
3	Metatheory	11
3.1	Confluence	11
3.2	Structural properties	13
3.3	Strong normalization of $\rightarrow_{\beta\wedge}$	16

3.4	Towards a generation principle for subtyping	18
3.4.1	Normal subtyping	19
3.5	Structural properties of NF_{λ}^{ω}	21
3.6	Equivalence of ordinary and normal subtyping	26
3.6.1	Least strict upper bound	28
3.7	A subtype checking algorithm, $AlgF_{\lambda}^{\omega}$	29
3.7.1	Example	30
3.8	Type checking and type reconstruction	30
3.9	Minimal typing	33
3.10	Subject reduction	36
4	Decidability	39
4.1	Decidability of context formation and kinding	39
4.2	Decidability of subtyping	40
4.2.1	Termination of subtype checking	40
4.3	Our decidability proof and full F_{\leq}	48
4.4	Decidability of type checking and type inference	48
5	Conclusions	50
6	Acknowledgements	51

1 Introduction

1.1 Motivation

The formal study of subtyping in programming languages was begun by Reynolds [50] and Cardelli [13], who used a lambda calculus with subtyping to model the refinement of interfaces in object-oriented languages. This led to a considerable body of work, modeling an increasing range of object-oriented features by combining subtyping with other type-theoretic constructs, including polymorphic functions [18, 34, 10], records with update and extension operators [13, 17], recursive types [2, 11]. F_{\leq} , a second-order λ -calculus with bounded quantification, was studied in [37] and [47]. Higher-order generalizations of subtyping appear in [12, 31, 45, 11]. Treating the interaction between interface refinement and encapsulation of objects in object-oriented programming has required higher-order generalizations of subtyping: the F-bounded quantification of Canning, Cook, Hill, Olthoff and Mitchell [12] or system F_{\leq}^{ω} [14, 16, 15, 45, 11]. Pure Type Systems with subtyping are studied in [53]; in particular an extension of F^{ω} with records, a fixed-point combinator, subtyping, and existential types is used to encode objects and shown to capture key object-oriented programming features such as aggregation, encapsulation, subtyping, self-reference, and late binding.

Type systems with subtyping have also arisen from the study of lambda calculi with intersection types at the University of Torino [33, 7] and elsewhere (see [19] for background and further references). Most of this work has been carried out in the setting of pure lambda calculi, but it has also been applied to programming language design by Reynolds [51]. Work combining intersection types with other typing disciplines can be found in [20], and a second-order λ -calculus with intersection types was studied in [47].

The system we study here, F_{λ}^{ω} (F-omega-meet), was first introduced in [30], where it was shown to be rich enough to provide a typed model of object-oriented programming with multiple inheritance. F_{λ}^{ω} is an extension of Girard's F^{ω} [39] with bounded quantification and intersection types, which can be seen as a natural generalization of the type disciplines present in the current literature, for example in [34, 47, 49, 21].

We use the definition of F_{λ}^{ω} from [27, 28] that differs from the one presented in [30] by introducing a richer notion of reduction on types, and thereby the four distributivity rules become particular cases of the conversion rule. This alternative, equivalent presentation provides a different view of the system that is the key to establishing the metatheory of the system.

1.2 Metatheory of F_{\wedge}^{ω}

We prove that the reduction relation of F_{\wedge}^{ω} is Church-Rosser. The reduction rules of F_{\wedge}^{ω} can be divided into two main groups, reductions on types ($\rightarrow_{\beta\wedge}$) and reductions on terms ($\rightarrow_{\beta\text{fors}}$). Although confluence is not a modular property in general, in our case it is possible to provide a modular proof of it. In section 3.1, we combine the independent proofs of confluence for reductions on types and confluence for reduction on terms to yield a proof of confluence of the reduction relation in the whole system. We also prove that the reduction on types ($\rightarrow_{\beta\wedge}$) is strongly normalizing (section 3.3) using a saturated sets model, together with other properties of the reduction relation.

Another property of interest is Subject Reduction (section 3.10), that the evaluation of a term preserves its type. Our approach is to give alternative, algorithmic presentations of the system; because these presentations are deterministic, they provide a powerful alternative induction principle with which to study the system.

In a type system with subtyping terms can have many different types. We show that in F_{\wedge}^{ω} the set of all the types of a term has a minimal type with respect to the subtyping relation (section 3.9).

1.3 Alternative presentations of the subtyping relation

Considering distributivity rules as part of the reduction relation on types suggests that to study the subtyping relation it is enough to concentrate on types in normal form. Note that the solution cannot be as simple as to restrict the subtyping rules of F_{\wedge}^{ω} to handle only types in normal form and replace conversion by reflexivity. The following is a good example of the problem to be solved. Consider the context

$$\Gamma \equiv W:K, X \leq \lambda Y:K.Y:K \rightarrow K, Z \leq X:K \rightarrow K,$$

where W has kind K and where X and Z are subtypes of the identity on K . Then $\Gamma \vdash X(ZW) \leq W$ is not derivable without using conversion, i.e. without performing any β -reduction, even when the conclusion is in normal form. (For a derivation see section 3.4.1.)

The subtyping rules of F_{\wedge}^{ω} are not syntax directed, in the sense that the form of a derivable subtyping judgement does not uniquely determine the last rule of its derivation, i.e. there might be more than one derivation of the same subtyping judgement. To develop a deterministic decision procedure to check subtyping, we need a new presentation of the subtyping relation that provides the foundations for a deterministic subtype-checking algorithm.

We develop a *normal subtyping system*, NF_{\wedge}^{ω} , in which only types in normal form are considered. We prove that derivations in NF_{\wedge}^{ω} can be normalized by eliminating transitivity and simplifying reflexivity. This simplification yields an algorithmic presentation, $AlgF_{\wedge}^{\omega}$. Moreover, we prove that $AlgF_{\wedge}^{\omega}$ is indeed an alternative presentation of the F_{\wedge}^{ω} subtyping relation, that is $\Gamma \vdash S \leq T$ if and only if $\Gamma^{nf} \vdash_{Alg} S^{nf} \leq T^{nf}$ (proposition 3.74).

These intermediate representations are instrumental in establishing several metatheoretic results including Subject Reduction and Decidability.

Martín Abadi and Luca Cardelli used these techniques in their book *A Theory of Objects* [1], demonstrating that our method extends naturally to a higher-order system with recursive types and object constructors. This work also influenced the work of Kathleen Fisher, who used this method in her dissertation (chapter 7 of [36]) to study a typed object calculus.

1.4 Decidability of F_{\wedge}^{ω}

Lambda calculi syntax is often presented by several interdependent relations or judgements, for example context formation, kinding, and typing. We say that a given lambda calculus is decidable if its typing relation is, but because of the interdependence of judgements this actually means the decidability of all the interwoven relations.

Our system, F_{\wedge}^{ω} , is a higher-order lambda calculus, which means that the sublanguage of types is a first-order lambda calculus containing lambda abstraction and application, in addition to intersections. Therefore the form of a type can change under reduction: in particular, a lambda abstraction on types can become an intersection type after reduction. Because F_{\wedge}^{ω} has reduction on types, we introduce a conversion rule that includes inter-convertible types in the subtype relation.

In this paper we give a positive answer to the decidability of typing in the presence of $\beta\wedge$ -convertible types and subtyping. We prove that subtyping in F_{\wedge}^{ω} is decidable, which *a fortiori* gives the decidability of subtyping for the F_{\leq}^{ω} fragment because the former is a conservative extension of the latter (namely, each subtyping judgement derivable in F_{\wedge}^{ω} containing no intersections other than the empty ones is also derivable in F_{\leq}^{ω}). A major task in establishing the decidability result for our system is proving that subtyping is decidable.

We establish the decidability of subtyping in F_{\wedge}^{ω} by proving that the algorithm described by $AlgF_{\wedge}^{\omega}$ terminates, which is equivalent to showing that the definition of the $AlgF_{\wedge}^{\omega}$ is well-founded. We discuss this further in section 4.2.1.

In the algorithm, checking whether $\Gamma \vdash_{Alg} ST \leq A$ is reduced to checking if $\Gamma \vdash_{Alg} lub_{\Gamma}(ST)^{nf} \leq A$, where $lub_{\Gamma}(ST)$ substitutes the leftmost innermost variable of ST by its bound in Γ . Such replacements may produce a term that is not in normal form, in which case we normalize it afterwards. (In an alternative notation used in [3], $\Gamma \vdash_{Alg} XS_1 \cdots S_n \leq A$ is reduced to checking if $\Gamma \vdash_{Alg} (\Gamma(X)S_1 \cdots S_n)^{nf} \leq A$.) The main problem here is that the size of the types to be examined in the recursive call may not decrease. This indicates that the proof of termination of the algorithm is not immediate. In particular, the proof of termination for the second-order calculus presented in [21] cannot be modified to serve our purposes, because of the interaction between reduction and the substitution of type variables by their bounds in our system. See section 4.2.1 for more details.

Typing and subtyping are defined using context formation and kinding judgements, so we show that the latter are decidable (section 4.1).

Finally we prove in section 4.4 that typing is decidable. To illustrate the problem of typing consider the following situation. Imagine trying to type fa in different contexts. We know we can type an application if we can find an arrow type for f whose domain is the type of a . In the context $\Gamma, X \leq S \rightarrow T : \star, f : X, a : S$, we need to use that the type X , with which f is declared, is a subtype of $S \rightarrow T$. Note that X , the minimal type of f , does not have enough structural information to type an application. Therefore, one needs to “climb” the subtyping hierarchy until a type with enough information is found, in this case an arrow type.

If we want to type fa with respect to the context $\Gamma' \equiv \Gamma, Z \leq \Lambda Y : \star . S \rightarrow Y : \star \rightarrow \star, W \leq Z : \star \rightarrow \star, f : WT, a : S$, we have to use that WT is a subtype of $S \rightarrow T$, which involves replacing W by Z , then replacing Z by $\Lambda Y : \star . S \rightarrow Y$ in WT , and finishing with a step of β -reduction. Since we need to continue making replacements and reduction steps until we find an arrow type, this procedure is slightly more complicated than finding the $lub_{\Gamma'}(WT)$, which is ZT . The situation is even more convoluted because of the presence of intersection types, but we leave that for section 4.4. This suggests that proving the termination of typechecking is not immediate, and that we will need to use similar ideas to those needed to prove the termination of subtype checking.

1.5 Other decidability proofs

The system F_{\leq} , a second-order λ -calculus with bounded quantification, was studied in [37]. In [47] it was proved that typing in F_{\leq} was undecidable and that undecidability was caused by the subtyping relation, the rule for bounded quantification being responsible for the failure.

An alternative rule for subtyping quantified types was presented in [21] in an attempt to find a decidable system, and the decidability of subtyping was proved for an extension of system F with bounded polymorphism, different from F_{\leq} . Unfortunately, the typing relation of [21] fails to satisfy the minimal types property. This failure, discovered by Ghelli, introduces serious problems in type checking and type inference. At the moment it is not clear how to solve them or, even more problematic, whether the typing relation is decidable. (See [22, 38] for more details.) The solution

we chose here to overcome this problem is to replace the subtyping rule between quantifiers by the corresponding rule of Cardelli and Wegner’s kernel Fun [18].

In [48] Steffen and Pierce studied F_{\leq}^{ω} (F-omega-sub), proving that typing is decidable and that the system satisfies the minimal types property. A central result in the proof of decidability is establishing the decidability of subtyping, a result first proved by Compagnoni in [25]. There are several differences between our work and theirs. Our results are for a stronger system which also includes intersection types, and our proof of decidability of subtyping predates theirs. A major technical difference is the choice of the intermediate subtyping system. We define the normal system NF_{\wedge}^{ω} which provides a generation principle for subtyping, yielding the algorithm $AlgF_{\wedge}^{\omega}$. In [48] the intermediate system, called a reducing system, leads to a more complex proof which involves several notions of reduction and three further refinements of the intermediate system.

A subtyping extension of the system λP , an abstract version of the type system of the Edinburgh Logical Framework LF, is studied in [4]. The proof of decidability is more involved because of the interdependency between the subtyping and typing judgements. The proof of decidability follows a strategy similar to the one presented here, where an alternative formulation of the system is shown to be an equivalent and terminating algorithm.

The calculus studied in [29] is a higher-order lambda calculus extending F_{\leq}^{ω} with bounded operator abstraction, in other words, the lambda abstraction on types has a bounded argument. These bounds introduce an interdependency between subtyping and kinding similar to the one in [4] between subtyping and typing. A radically different approach to proving decidability is used, where a typed operational semantics following [40, 41] is defined and proved equivalent to the original system using a Kripke model, and termination is proved by induction on derivations in the typed operational semantics.

A higher-order typed lambda calculus with subtyping, bounded quantification, and datatypes with built in subtyping and overloading (constructor subtyping) is defined and studied in [8]. The proof of decidability is a straightforward adaptation of the proof presented here. In [9], an extension of the Calculus of Constructions with constructor subtyping is presented. To avoid the natural interdependence between typing and subtyping, the subtyping relation is defined on pseudoterms (terms that may or may not be typable), as in [53, 23]. The price for this simplification is the need for an unnatural subtyping rule comparing untypable terms (the application of a quantified term to a term).

1.6 Results

- We define the typed lambda calculus F_{\wedge}^{ω} , a natural generalization of Girard’s system F^{ω} with intersection types and bounded polymorphism. A novel aspect of our presentation is the use of term rewriting techniques to present intersection types, which clearly splits the computational semantics (reduction rules) from the syntax (inference rules) of the system.
- We prove that the reduction relation of F_{\wedge}^{ω} is confluent with a modular proof that combines the confluence property of the reduction on types and the reduction on terms.
- We prove the strong normalization property of $\rightarrow_{\beta_{\wedge}}$ on well-formed types.
- We define a normalized system NF_{\wedge}^{ω} equivalent to the original presentation of subtyping, and prove the transitivity elimination and reflexivity simplification properties.
- We define a subtyping algorithm $AlgF_{\wedge}^{\omega}$, and prove that it is equivalent to the original presentation.
- In section 3.8, we prove that F_{\wedge}^{ω} satisfies the minimal types property, and we provide an algorithm for computing minimal types.
- We prove that F_{\wedge}^{ω} satisfies the subject reduction property using the minimal types property.
- We prove that the subtyping relation in F_{\wedge}^{ω} is decidable and that type inference in F_{\wedge}^{ω} is decidable.

- We present the first proof of decidability of subtyping of a higher-order lambda calculus.

2 Syntax of F_{\wedge}^{ω}

We now present the syntax of F_{\wedge}^{ω} : its syntactic categories and judgements.

2.1 Syntactic categories

The syntactic categories of F_{\wedge}^{ω} are kinds, types, terms, and contexts.

The *kinds* of F_{\wedge}^{ω} are those of F^{ω} : the kind \star of proper types and the kinds $K_1 \rightarrow K_2$ of functions on types (sometimes called type operators).

$$\begin{array}{l} \mathbb{K} ::= \star \quad \text{types} \\ | \quad \mathbb{K} \rightarrow \mathbb{K} \quad \text{type operators} \end{array}$$

The language of *types* of F_{\wedge}^{ω} is a straightforward higher-order extension of F_{\leq} , Cardelli and Wegner’s second-order calculus of bounded quantification. Like F_{\leq} , it includes type variables (written X), function types ($T \rightarrow T'$), and polymorphic types ($\forall X \leq \bar{T} : K.T'$), in which the bound type variable X ranges over all subtypes of the upper bound T . Moreover, like F^{ω} , we allow types to be abstracted on types ($\Lambda X : K.T$) and applied to argument types ($T T'$); in effect, these forms introduce a simply typed λ -calculus at the level of types. Finally, we allow arbitrary finite intersections ($\bigwedge^K [T_1..T_n]$), where all the T_i ’s are members of the same kind K .

$$\begin{array}{l} \mathbb{T} ::= X \quad \text{type variable} \\ | \quad \mathbb{T} \rightarrow \mathbb{T} \quad \text{function type} \\ | \quad \forall X \leq \mathbb{T} : \mathbb{K}. \mathbb{T} \quad \text{polymorphic type} \\ | \quad \Lambda X : \mathbb{K}. \mathbb{T} \quad \text{operator abstraction} \\ | \quad \mathbb{T} \mathbb{T} \quad \text{operator application} \\ | \quad \bigwedge^K [\mathbb{T}.. \mathbb{T}] \quad \text{intersection at kind } \mathbb{K} \end{array}$$

We use the abbreviation \top^K for nullary intersections and sometimes $X : K$ for $X \leq \top^K : K$.

$$\top^K \equiv \bigwedge^K [] \quad X : K \equiv X \leq \top^K : K$$

We drop the maximal type *Top* of F_{\leq} , since its role is played here by the empty intersection \top^* . For technical convenience, we provide kind annotations on bound variables and intersections so that every type has an “obvious kind,” which can be read off directly from its structure and the kind declarations in the context.

The language of terms includes the variables (x), applications ($e e$), and functional abstractions ($\lambda x : T.e$) of the simply typed λ -calculus, plus the type abstraction ($\lambda X \leq T : K.e$) and application ($e T$) of F^{ω} . As in F_{\leq} , each type variable is given an upper bound at the point where it is introduced.

Intersection types are introduced by expressions of the form “for($X \in T_1..T_n$) e ”, which can be read as instructions to the type-checker to analyze the expression e separately under the assumptions $X \equiv T_1, X \equiv T_2, \dots, X \equiv T_n$ and conjoin the results. For example, if $+ : \text{Int} \rightarrow \text{Int} \rightarrow \text{Int} \wedge \text{Real} \rightarrow \text{Real} \rightarrow \text{Real}$, then we can derive:

$$\text{for}(X \in \text{Int}, \text{Real}) \lambda x : X. x + x \quad : \quad \text{Int} \rightarrow \text{Int} \wedge \text{Real} \rightarrow \text{Real}.$$

$$\begin{array}{l} e ::= x \quad \text{variable} \\ | \quad \lambda x : \mathbb{T}. e \quad \text{abstraction} \\ | \quad e e \quad \text{application} \\ | \quad \lambda X \leq \mathbb{T} : \mathbb{K}. e \quad \text{type abstraction} \\ | \quad e \mathbb{T} \quad \text{type application} \\ | \quad \text{for}(X \in \mathbb{T}.. \mathbb{T}) e \quad \text{alternation} \end{array}$$

The operational semantics of F_{\wedge}^{ω} is given by the following reduction rules on types and terms.

DEFINITION 2.1 (*Reduction rules for types*)

1. $(\Lambda X:K.T_1)T_2 \rightarrow_{\beta\wedge} T_1[X \leftarrow T_2]$
2. $S \rightarrow \wedge^*[T_1..T_n] \rightarrow_{\beta\wedge} \wedge^*[S \rightarrow T_1 .. S \rightarrow T_n]$
3. $\forall X \leq S:K. \wedge^*[T_1..T_n] \rightarrow_{\beta\wedge} \wedge^*[\forall X \leq S:K.T_1 .. \forall X \leq S:K.T_n]$
4. $\Lambda X:K_1. \wedge^{K_2}[T_1..T_n] \rightarrow_{\beta\wedge} \wedge^{K_1 \rightarrow K_2}[\Lambda X:K_1.T_1 .. \Lambda X:K_1.T_n]$
5. $(\wedge^{K_1 \rightarrow K_2}[T_1..T_n])U \rightarrow_{\beta\wedge} \wedge^{K_2}[T_1 U .. T_n U]$
6. $\wedge^K[T_1 .. \wedge^K[S_1..S_n] .. T_m] \rightarrow_{\beta\wedge} \wedge^K[T_1 .. S_1..S_n .. T_m]$

The first rule is the usual β -reduction rule for types. Rules 2 through 5 express the fact that intersections in positive positions distribute with respect to the other type constructors. Rule 6 states that intersection is an associative operator. In section 3.3 we consider the reduction defined by rules 1 through 5 as $\rightarrow_{\beta\wedge}$ and the one defined by 6 as \rightarrow_a (a comes from associativity). The left-hand side of each reduction rule is a *redex* and the right-hand side its *reduct*. The relation $\rightarrow_{\beta\wedge}$ is extended so as to become a compatible relation with respect to type formation, $\twoheadrightarrow_{\beta\wedge}$ is the transitive and reflexive closure of $\rightarrow_{\beta\wedge}$, and $=_{\beta\wedge}$ is the least equivalence relation containing $\rightarrow_{\beta\wedge}$. The capture-avoiding substitution of S for X in T is written $T[X \leftarrow S]$. Substitution is written similarly for terms, and is extended point-wise to contexts. The $\beta\wedge$ -normal form of a type S is written S^{nf} , and is extended point-wise to contexts.

2.2 Contexts

A *context* Γ is a finite sequence of typing and subtyping assumptions for a set of term and type variables.

The empty context is written \emptyset . Term variable bindings have the form $x:T$; type variable bindings have the form $X \leq T:K$, where T is the upper bound of X and K is the kind of T .

$\Gamma ::= \emptyset$	empty context
$\Gamma, x:T$	term variable declaration
$\Gamma, X \leq T:K$	type variable declaration

When writing nonempty contexts, we omit the initial \emptyset . The domain of Γ is written $\text{dom}(\Gamma)$. The functions $\text{FV}(_)$ and $\text{FTV}(_)$ give the sets of free term variables and free type variables of a term, type, or context. Since we are careful to ensure that no variable is bound more than once, we sometimes abuse notation and consider contexts as finite functions: $\Gamma(X)$ yields the bound of X in Γ , where X is implicitly asserted to be in $\text{dom}(\Gamma)$.

Types, terms, contexts, judgements, and derivations that differ only in the names of bound variables are considered identical. The underlying idea is that variables are de Bruijn indexes [35].

DEFINITION 2.2 (*Closed*)

1. A term e is closed with respect to a context Γ if $\text{FV}(e) \cup \text{FTV}(e) \subseteq \text{dom}(\Gamma)$.
2. A type T is closed with respect to a context Γ if $\text{FTV}(T) \subseteq \text{dom}(\Gamma)$.
3. A typing judgement $\Gamma \vdash e : T$ is closed if e and T are closed with respect to Γ .
4. A kinding judgement $\Gamma \vdash T : K$ is closed if T is closed with respect to Γ .
5. A subtyping judgement $\Gamma \vdash S \leq T$ is closed if S and T are closed with respect to Γ .

We consider only closed typing judgements. Observe that in the limit case of the rule T-MEET, when $n = 0$, not having the closure convention would allow nonsensical terms to be typed. On the other hand, the free variable lemma (lemma 3.17) guarantees that kinding judgements are closed and the well-kindedness of subtyping (lemma 3.32) ensures that subtyping judgements are closed as well.

DEFINITION 2.3 (*Reduction rules for terms*)

1. $(\lambda x:T_1.e_1)e_2 \rightarrow_{\beta fors} e_1[x \leftarrow e_2]$
2. $(\lambda X \leq T_1:K_1.e)T \rightarrow_{\beta fors} e[X \leftarrow T]$
3. $(\text{for}(X \in T_1..T_n)e_1)e_2 \rightarrow_{\beta fors} \text{for}(X \in T_1..T_n)(e_1 e_2)$
4. $\text{for}(X \in T_1..T_n)e \rightarrow_{\beta fors} e$, if $X \notin \text{FTV}(e)$

Rules 1 and 2 are the β -reductions on terms. Rule 3 says that the for constructor can be pushed to the outermost level. We consider the reduction defined by rules 1 through 3 as $\rightarrow_{\beta for}$ and the one defined by 4 as \rightarrow_s (s comes from simplification). The left-hand side of each reduction rule is a *redex* and the right-hand side its *reduct*. The relation $\rightarrow_{\beta fors}$ is extended so as to become a compatible relation with respect to term formation, $\twoheadrightarrow_{\beta fors}$ is the transitive reflexive closure of $\rightarrow_{\beta fors}$, and $=_{\beta fors}$ is the least equivalence relation containing $\rightarrow_{\beta fors}$.

The rules of F_{\wedge}^{ω} are organized as proof systems for four interdependent judgement forms:

- $\Gamma \vdash \text{ok}$ well-formed context
- $\Gamma \vdash T : K$ well-kinded type
- $\Gamma \vdash S \leq T$ subtype
- $\Gamma \vdash e : T$ well-typed term.

We sometimes use the metavariable Σ to range over statements (right-hand sides of judgements) of any of these four forms.

2.3 Context formation

The rules for well-formed contexts are the usual ones: a start rule for the empty context and rules allowing a given well-formed context to be extended with either a term variable binding or a type variable binding.

$$\begin{array}{l} \emptyset \vdash \text{ok} \qquad \qquad \qquad \text{(C-EMPTY)} \\ \frac{\Gamma \vdash T : \star \quad x \notin \text{dom}(\Gamma)}{\Gamma, x:T \vdash \text{ok}} \qquad \qquad \qquad \text{(C-VAR)} \\ \frac{\Gamma \vdash T : K \quad X \notin \text{dom}(\Gamma)}{\Gamma, X \leq T:K \vdash \text{ok}} \qquad \qquad \qquad \text{(C-TVAR)} \end{array}$$

2.4 Type formation

For each type constructor, we give a rule specifying how it can be used to build well-formed type expressions. The rules K-OABS and K-OAPP form type abstractions and type applications (essentially as in a simply typed λ -calculus), the rule K-MEET constructs intersection types, and aside from some extra kinding information the other rules are the same as in the second order case.

The well-formedness premise $\Gamma \vdash \text{ok}$ in K-MEET (and in T-MEET below) is required for the case where $n = 0$.

$$\frac{\Gamma_1, X \leq T:K, \Gamma_2 \vdash \text{ok}}{\Gamma_1, X \leq T:K, \Gamma_2 \vdash X : K} \qquad \qquad \qquad \text{(K-TVAR)}$$

$$\begin{array}{c}
\frac{\Gamma \vdash T_1 : \star \quad \Gamma \vdash T_2 : \star}{\Gamma \vdash T_1 \rightarrow T_2 : \star} \quad (\text{K-ARROW}) \\
\frac{\Gamma, X \leq T_1 : K_1 \vdash T_2 : \star}{\Gamma \vdash \forall X \leq T_1 : K_1. T_2 : \star} \quad (\text{K-ALL}) \\
\frac{\Gamma, X : K_1 \vdash T_2 : K_2}{\Gamma \vdash \Lambda X : K_1. T_2 : K_1 \rightarrow K_2} \quad (\text{K-OABS}) \\
\frac{\Gamma \vdash S : K_1 \rightarrow K_2 \quad \Gamma \vdash T : K_1}{\Gamma \vdash ST : K_2} \quad (\text{K-OAPP}) \\
\frac{\Gamma \vdash \text{ok} \quad \text{for each } i \in \{1..n\}, \Gamma \vdash T_i : K}{\Gamma \vdash \bigwedge^K [T_1..T_n] : K} \quad (\text{K-MEET})
\end{array}$$

2.5 Subtyping

The rules defining the subtype relation are a natural extension of familiar calculi of bounded quantification. Aside from some extra well-formedness conditions, the rules S-TRANS, S-TVAR, and S-ARROW are the same as in the usual, second-order case. Rules S-OABS and S-OAPP extend the subtype relation point-wise to kinds other than \star . The rule of type conversion in F^{ω} , that is, if $\Gamma \vdash e : T$ and $T =_{\beta} T'$ then $\Gamma \vdash e : T'$, is captured here as the subtyping rule S-CONV, which also gives reflexivity as a special case. The rule S-ALL is the rule of Cardelli and Wegner's language *Fun* [18] in which the bounds of the quantifiers are equal. Rules S-MEET-G and S-MEET-LB specify that an intersection of a set of types is the set's order-theoretic greatest lower bound, with respect to the subtyping order.

$$\begin{array}{c}
\frac{\Gamma \vdash S : K \quad \Gamma \vdash T : K \quad S =_{\beta \wedge} T}{\Gamma \vdash S \leq T} \quad (\text{S-CONV}) \\
\frac{\Gamma \vdash S \leq T \quad \Gamma \vdash T \leq U}{\Gamma \vdash S \leq U} \quad (\text{S-TRANS}) \\
\frac{\Gamma_1, X \leq T : K, \Gamma_2 \vdash \text{ok}}{\Gamma_1, X \leq T : K, \Gamma_2 \vdash X \leq T} \quad (\text{S-TVAR}) \\
\frac{\Gamma \vdash T_1 \leq S_1 \quad \Gamma \vdash S_2 \leq T_2 \quad \Gamma \vdash S_1 \rightarrow S_2 : \star}{\Gamma \vdash S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2} \quad (\text{S-ARROW}) \\
\frac{\Gamma, X \leq U : K \vdash S \leq T \quad \Gamma \vdash \forall X \leq U : K. S : \star}{\Gamma \vdash \forall X \leq U : K. S \leq \forall X \leq U : K. T} \quad (\text{S-ALL}) \\
\frac{\Gamma, X : K \vdash S \leq T}{\Gamma \vdash \Lambda X : K. S \leq \Lambda X : K. T} \quad (\text{S-OABS}) \\
\frac{\Gamma \vdash S \leq T \quad \Gamma \vdash SU : K}{\Gamma \vdash SU \leq TU} \quad (\text{S-OAPP}) \\
\frac{\text{for each } i \in \{1..n\}, \Gamma \vdash S \leq T_i \quad \Gamma \vdash S : K}{\Gamma \vdash S \leq \bigwedge^K [T_1..T_n]} \quad (\text{S-MEET-G}) \\
\frac{\Gamma \vdash \bigwedge^K [T_1..T_n] : K}{\Gamma \vdash \bigwedge^K [T_1..T_n] \leq T_i} \quad (\text{S-MEET-LB})
\end{array}$$

2.6 Term formation

Except for T-MEET and T-FOR, the term formation rules are precisely those of the second-order calculus of bounded quantification. T-FOR provides for type checking under any of a set of alternate assumptions. For each S_i , the type derived for the instance of the body e when X is replaced by

S_i is a valid type of the for expression itself. The T-MEET rule can then be used to collect these separate typings into a single intersection. Type-theoretically, T-MEET is the introduction rule for the \wedge constructor; the corresponding elimination rule need not be given explicitly, since it follows from T-SUBSUMPTION and S-MEET-LB.

$$\frac{\Gamma_1, x:T, \Gamma_2 \vdash \text{ok}}{\Gamma_1, x:T, \Gamma_2 \vdash x : T} \quad (\text{T-VAR})$$

$$\frac{\Gamma, x:T_1 \vdash e : T_2}{\Gamma \vdash \lambda x:T_1. e : T_1 \rightarrow T_2} \quad (\text{T-ABS})$$

$$\frac{\Gamma \vdash f : T_1 \rightarrow T_2 \quad \Gamma \vdash a : T_1}{\Gamma \vdash f a : T_2} \quad (\text{T-APP})$$

$$\frac{\Gamma, X \leq T_1 : K_1 \vdash e : T_2}{\Gamma \vdash \lambda X \leq T_1 : K_1. e : \forall X \leq T_1 : K_1. T_2} \quad (\text{T-TABS})$$

$$\frac{\Gamma \vdash f : \forall X \leq T_1 : K_1. T_2 \quad \Gamma \vdash S \leq T_1}{\Gamma \vdash f S : T_2[X \leftarrow S]} \quad (\text{T-TAPP})$$

$$\frac{\Gamma \vdash e[X \leftarrow S] : T \quad S \in \{S_1 \dots S_n\}}{\Gamma \vdash \text{for}(X \in S_1 \dots S_n) e : T} \quad (\text{T-FOR})$$

$$\frac{\Gamma \vdash \text{ok} \quad \text{for each } i \in \{1..n\}, \Gamma \vdash e : T_i}{\Gamma \vdash e : \bigwedge^* [T_1 \dots T_n]} \quad (\text{T-MEET})$$

$$\frac{\Gamma \vdash e : S \quad \Gamma \vdash S \leq T}{\Gamma \vdash e : T} \quad (\text{T-SUBSUMPTION})$$

Most of the rules include premises which have two rather different sorts: *structural premises*, which play an essential role in giving the rule its intended semantic force, and *well-formation premises*, which ensure that the entities named in the rule are of the expected sorts.

We sometimes omit well-formation premises that can be derived from others. For example, in the rule S-ARROW, we drop the premise $\Gamma \vdash T_1 \rightarrow T_2 : \star$, since it follows from $\Gamma \vdash S_1 \rightarrow S_2 : \star$ using the properties proved in section 3.2.

2.7 Discussion

An equivalent presentation of intersection types uses binary intersections as in [32]. The intersection of S and T is then written $S \wedge T$, and there is a maximal element at each kind, ω^K . The rules of the system have to be modified according to this alternative notation. In most cases, each of our rules about intersection types has to be replaced by two rules, one for the binary case and another for the maximal element. For example, the reduction rule

$$\forall X \leq S : K. \bigwedge^* [T_1 \dots T_n] \rightarrow_{\beta \wedge} \bigwedge^* [\forall X \leq S : K. T_1 \dots \forall X \leq S : K. T_n]$$

is replaced by

$$\begin{aligned} \forall X \leq S : K. T_1 \wedge T_2 &\rightarrow_{\beta \wedge} \forall X \leq S : K. T_1 \wedge \forall X \leq S : K. T_2 \quad \text{and} \\ \forall X \leq S : K. \omega^* &\rightarrow_{\beta \wedge} \omega^*. \end{aligned}$$

Similar replacement takes place for rules 3 through 5 in definition 2.1. The term formation rule K-MEET is replaced by the two following rules.

$$\frac{\Gamma \vdash S : K \quad \Gamma \vdash T : K}{\Gamma \vdash S \wedge T : K} \quad (\text{K-INT})$$

$$\frac{\Gamma \vdash \text{ok}}{\Gamma \vdash \omega^K : K} \quad (\text{K-MAX})$$

The rule S-MEET-G is replaced by the following two rules.

$$\frac{\Gamma \vdash S \leq T_1 \quad \Gamma \vdash S \leq T_2}{\Gamma \vdash S \leq T_1 \wedge T_2} \quad (\text{S-INT-G})$$

$$\frac{\Gamma \vdash S : K}{\Gamma \vdash S \leq \omega^K} \quad (\text{S-MAX})$$

For those familiar with the λ -cube [5], F^ω corresponds to λ_ω , the system defined by the rules (\star, \star) , (\square, \star) , and (\square, \square) . If K is a kind defined by the grammar \mathbb{K} , then

$$\Gamma \vdash_{\lambda_\omega} K : \square.$$

The rule (\square, \square) corresponds to the recursive step in the definition of \mathbb{K} ; the rule (\star, \star) corresponds to K-ARROW, and K-ALL is the parallel of rule (\square, \star) enriched with subtyping.

3 Metatheory

3.1 Confluence

In this section, we show that the system F_\wedge^ω is confluent. By that we mean that the reduction $\rightarrow_{\beta_{fors}} \cup \rightarrow_{\beta_\wedge}$ defined by putting together the reduction on terms, $\rightarrow_{\beta_{fors}}$ (definition 2.3), and the reduction on types, $\rightarrow_{\beta_\wedge}$ (definition 2.1), satisfies the Church-Rosser property [24]. We use the Hindley-Rossen lemma (c.f. 3.3.5 [6] and [43]) to establish this result. This factors the proof into two parts:

1. proving that the reductions $\rightarrow_{\beta_{fors}}$ and $\rightarrow_{\beta_\wedge}$ commute, and
2. proving that the reductions $\rightarrow_{\beta_{fors}}$ and $\rightarrow_{\beta_\wedge}$ satisfy the Church-Rosser property.

Full details of the proofs of this section as well as intermediate results can be found in [27]. Remember that two binary relations \rightarrow_1 and \rightarrow_2 commute if given $A \rightarrow_1 B$ and $A \rightarrow_2 C$, there exists D such that $C \rightarrow_1 D$ and $B \rightarrow_2 D$. In order to prove that $\rightarrow_{\beta_{fors}}$ and $\rightarrow_{\beta_\wedge}$ commute we use the following lemma.

LEMMA 3.1 (3.3.6 [6]) Let \rightarrow_1 and \rightarrow_2 be two binary relations on a set X . Suppose that if $A \rightarrow_1 B$ and $A \rightarrow_2 C$, there exists D such that $C \rightarrow_{=1} D$ and $B \rightarrow_2 D$, where $\rightarrow_{=1}$ is the reflexive closure of \rightarrow_1 . Hence \rightarrow_1 and \rightarrow_2 commute.

LEMMA 3.2 If $A \rightarrow_{\beta_{fors}} B$ and $A \rightarrow_{\beta_\wedge} C$, there exists D such that $C \rightarrow_{=\beta_{fors}} D$ and $B \rightarrow_{\beta_\wedge} D$

PROOF: By induction on the structure of E . □

COROLLARY 3.3 $\rightarrow_{\beta_\wedge}$ and $\rightarrow_{\beta_{fors}}$ commute.

The Church-Rosser theorem for $\rightarrow_{\beta_\wedge}$

We now prove the Church-Rosser property for the reduction defined in 2.1. The strategy we use here is similar to the one used in chapter 11 section 1 of [6] to prove the corresponding result for \rightarrow_β in the type-free λ -calculus.

In order to prove the Church-Rosser property for $\rightarrow_{\beta_\wedge}$ it is sufficient to show the following *strip* lemma.

LEMMA 3.4 (Strip) Let S, T_1 , and $T_2 \in \mathbb{T}$. If $S \rightarrow_{\beta_\wedge} T_1$ and $S \rightarrow_{\beta_\wedge} T_2$, then there exists $T_3 \in \mathbb{T}$ such that $T_1 \rightarrow_{\beta_\wedge} T_3$ and $T_2 \rightarrow_{\beta_\wedge} T_3$.

The idea of the proof is as follows. Let T_1 be the result of replacing the redex R in S by its reduct R' . If we keep track of what happens with R during the reduction $S \rightarrow_{\beta_\wedge} T_2$, then we can find T_3 . To be able to trace R we define a new set of terms $\underline{\mathbb{T}}$ where redexes can appear underlined. Consequently, if we underline R in S we only need to reduce all occurrences of the underlined R in T_2 to obtain T_3 .

THEOREM 3.5 (*Church-Rosser for $\rightarrow_{\beta\wedge}$*)

If $S, T_1,$ and $T_2 \in \mathbb{T}$ are such that $S \twoheadrightarrow_{\beta\wedge} T_1$ and $S \twoheadrightarrow_{\beta\wedge} T_2$, then there exists $T_3 \in \mathbb{T}$ such that $T_1 \twoheadrightarrow_{\beta\wedge} T_3$ and $T_2 \twoheadrightarrow_{\beta\wedge} T_3$.

PROOF: By induction on the generation of $S \twoheadrightarrow_{\beta\wedge} T_1$. □

The Church-Rosser theorem for $\rightarrow_{\beta fors}$

Next we prove the Church-Rosser property for the reduction defined in definition 2.3.

THEOREM 3.6 (*Church-Rosser for $\rightarrow_{\beta fors}$*)

Let $e, f_1, f_2 \in \mathbb{E}$. If $e \twoheadrightarrow_{\beta fors} f_1$ and $e \twoheadrightarrow_{\beta fors} f_2$, then there exists $f_3 \in \mathbb{E}$ such that $f_1 \twoheadrightarrow_{\beta fors} f_3$ and $f_2 \twoheadrightarrow_{\beta fors} f_3$.

The idea of the proof is as follows. We prove that $\rightarrow_{\beta for}$ and \rightarrow_s are Church-Rosser (theorem 3.7 and lemma 3.8); that \rightarrow_s reduction steps can be postponed (lemma 3.9), and that $\twoheadrightarrow_{\beta for}$ and \twoheadrightarrow_s commute (lemma 3.10).

Those four results allow us to prove the Church-Rosser theorem for $\rightarrow_{\beta fors}$. Let $e, e_1, e_2 \in \mathbb{E}$, such that $e \twoheadrightarrow_{\beta fors} e_1$ and $e \twoheadrightarrow_{\beta fors} e_2$. Then, by s -postponement, there exist f_1 and f_2 ; by Church-Rosser for $\rightarrow_{\beta for}$, there exists f_3 ; and, by lemma 3.10, there exist f_4 and f_5 , and finally, by Church-Rosser for \rightarrow_s , there exists e_3 which completes the following diagram.

$$\begin{array}{ccccc}
 e & \xrightarrow{\beta for} & f_1 & \xrightarrow{s} & e_1 \\
 \beta for \downarrow & & \beta for \downarrow & & \beta for \downarrow \\
 f_2 & \xrightarrow{\beta for} & f_3 & \xrightarrow{s} & f_4 \\
 s \downarrow & & s \downarrow & & s \downarrow \\
 e_2 & \xrightarrow{\beta for} & f_5 & \xrightarrow{s} & e_3
 \end{array}$$

The Church-Rosser property for $\rightarrow_{\beta for}$ follows using the same strategy used to prove theorem 3.5.

THEOREM 3.7 (*Church-Rosser for $\rightarrow_{\beta for}$*)

If $e, f_1,$ and $f_2 \in \mathbb{E}$ are such that $e \twoheadrightarrow_{\beta for} f_1$ and $e \twoheadrightarrow_{\beta for} f_2$, then there exists $f_3 \in \mathbb{E}$ such that $f_1 \twoheadrightarrow_{\beta for} f_3$ and $f_2 \twoheadrightarrow_{\beta for} f_3$.

The Church-Rosser property for \rightarrow_s is proved using the Newman's proposition 3.1.25 in [6] from [46], by proving that \rightarrow_s is strongly normalizing and weak Church-Rosser.

LEMMA 3.8 (*Church-Rosser for \rightarrow_s*) If $e, e_1,$ and $e_2 \in \mathbb{E}$ are such that $e \twoheadrightarrow_s e_1$ and $e \twoheadrightarrow_s e_2$, then there exists e_3 such that $e_1 \twoheadrightarrow_s e_3$ and $e_2 \twoheadrightarrow_s e_3$.

LEMMA 3.9 (*s -postponement*) If $e \twoheadrightarrow_s e_1$ and $e_1 \twoheadrightarrow_{\beta for} e_2$, then there exists e_3 such that $e \twoheadrightarrow_{\beta for} e_3$ and $e_3 \twoheadrightarrow_s e_1$.

LEMMA 3.10 If $e, e_1,$ and $e_2 \in \mathbb{E}$ are such that $e \twoheadrightarrow_{\beta for} e_1$ and $e \twoheadrightarrow_s e_2$ then there exists e_3 such that $e_1 \twoheadrightarrow_s e_3$ and $e_2 \twoheadrightarrow_{\beta for} e_3$.

Finally, we can state and prove the confluence property for the reduction relation of F_{\wedge}^{ω} .

Confluence of F_{\wedge}^{ω}

THEOREM 3.11 (*Church-Rosser for $\rightarrow_{\beta fors} \cup \rightarrow_{\beta \wedge}$*)

If E, F , and $G \in \mathbb{T} \cup \mathbb{E}$ are such that $E \rightarrow_{\beta fors \cup \beta \wedge} F$ and $E \rightarrow_{\beta fors \cup \beta \wedge} G$, then there exists $H \in \mathbb{T} \cup \mathbb{E}$ such that $F \rightarrow_{\beta fors \cup \beta \wedge} H$ and $G \rightarrow_{\beta fors \cup \beta \wedge} H$.

PROOF: By the commutativity of $\rightarrow_{\beta fors}$ and $\rightarrow_{\beta \wedge}$ (corollary 3.3) and the Church-Rosser property of $\rightarrow_{\beta fors}$ and $\rightarrow_{\beta \wedge}$ (theorems 3.5 and 3.6). \square

The Church-Rosser theorem has interesting corollaries that we will use in the sequel.

COROLLARY 3.12 See chapter 3 of [6]. Let R be a reduction relation satisfying the Church-Rosser property. Then

1. If $T =_R S$, then there exists U such that $T \rightarrow_R U$ and $S \rightarrow_R U$.
2. If T is a normal form of S , then $S \rightarrow_R T$.
3. Each term has at most one R -normal form.

FACT 3.13

1. $\forall X \leq S: K.T =_{\beta \wedge} \top^*$ if and only if $T =_{\beta \wedge} \top^*$.
2. $\Lambda X: K.T =_{\beta \wedge} \top^*$ if and only if $T =_{\beta \wedge} \top^*$.
3. $S \rightarrow T =_{\beta \wedge} \top^*$ if and only if $T =_{\beta \wedge} \top^*$.
4. $T S =_{\beta \wedge} \top^*$ if and only if $T =_{\beta \wedge} \top^*$.

LEMMA 3.14 If $S \rightarrow_{\beta \wedge} S'$, then $S[X \leftarrow U] \rightarrow_{\beta \wedge} S'[X \leftarrow U]$.

3.2 Structural properties

This section establishes a number of structural properties of F_{\wedge}^{ω} . Except where noted, the proofs proceed by structural induction and are straightforward when performed in the order in which they appear.

LEMMA 3.15 If $\Gamma \vdash \Sigma$ and Γ_1 is a prefix of Γ , then $\Gamma_1 \vdash \text{ok}$ as a subderivation. Moreover, except for the case $\Gamma_1 \equiv \Gamma$ and $\Sigma \equiv \text{ok}$, the subderivation is strictly shorter.

LEMMA 3.16 (*Generation for context judgements*)

1. If $\Gamma_1, X \leq T: K, \Gamma_2 \vdash \text{ok}$, then $\Gamma_1 \vdash T : K$ by a proper subderivation.
2. If $\Gamma_1, x: T, \Gamma_2 \vdash \text{ok}$, then $\Gamma_1 \vdash T : \star$ by a proper subderivation.

LEMMA 3.17 (*Free variables*)

1. If $\Gamma \vdash T : K$, then $\text{FTV}(T) \subseteq \text{dom}(\Gamma)$.
2. If $\Gamma \vdash \text{ok}$, then each variable or type variable in $\text{dom}(\Gamma)$ is declared only once.

If one tries to prove Weakening (Corollary 3.21) directly by induction on derivations the induction hypothesis is too weak in the cases for K-ALL and S-OABS, for example. This problem occurs in the lambda calculus without subtyping for the abstraction rule, and was identified by McKinna and Pollack for Pure Type Systems. We adapt their idea of *renaming* [44].

DEFINITION 3.18 (*Parallel Substitution*) A *parallel substitution* γ for Γ is an assignment of types to type variables in $\text{dom}(\Gamma)$ and terms to term variables in $\text{dom}(\Gamma)$. A *renaming* for Γ in Δ is a parallel substitution γ from variables to variables such that

- for every $x:A$ in Γ , $\gamma(x):A[\gamma]$ is in Δ , and
- for every $X \leq T : K$ in Γ , $\gamma(X) \leq A[\gamma] : K$ is in Δ .

We write $\Sigma[\gamma]$ for the result of performing the substitution γ in the judgement Σ . The renaming $\gamma\{x \mapsto y\}$ maps x to y and behaves like γ elsewhere, similarly for type variables.

LEMMA 3.19 (*Renaming*) If $\Delta \vdash \text{ok}$ and γ is a renaming for Γ in Δ then $\Gamma \vdash \Sigma$ implies $\Delta \vdash \Sigma[\gamma]$.

PROOF: By induction on the derivation of $\Gamma \vdash \Sigma$. Most cases follow easily using the induction hypothesis or the definition of renaming. We illustrate here the case for K-ALL, which is representative of the interesting cases.

Let $Z \notin \text{dom}(\Delta)$. Define γ_0 as $\gamma_0 \equiv \gamma\{X \mapsto Z\}$, then γ_0 is a renaming for Γ , $X \leq T_1 : K_1$ in Δ , $Z \leq T_1[\gamma_0] : K_1$. By lemmas 3.15 and 3.16(1), there exists a shorter subderivation of $\Gamma \vdash T_1 : K_1$, and by the free variables lemma (lemma 3.17), $X \notin \text{FV}(T_1)$, therefore $T_1[\gamma_0] \equiv T_1[\gamma]$.

We need to show that $\Delta, Z \leq T_1[\gamma] : K_1 \vdash \text{ok}$. By assumption we know that $\Delta \vdash \text{ok}$, by the induction hypothesis, $\Delta \vdash T_1[\gamma] : K_1$. Since we chose Z not to be in $\text{dom}(\Delta)$, by K-TVAR, $\Delta, Z \leq T_1[\gamma] : K_1 \vdash \text{ok}$.

We can now apply the induction hypothesis to prove $\Delta, Z \leq T_1[\gamma] : K_1 \vdash T_2[\gamma_0] : \star$. By K-ALL, $\Delta \vdash \forall Z \leq T_1[\gamma] : K_1. T_2[\gamma_0] : \star$, and by the definition of substitution $\Delta \vdash (\forall X \leq T_1 : K_1. T_2 : \star)[\gamma]$. \square

Weakening now follows as a corollary of renaming taking γ to be the identity substitution.

DEFINITION 3.20 (*Context inclusion*) Let Γ and Δ be contexts. $\Gamma \subseteq \Delta$ if for all $x : \mathbb{T} \in \Gamma$ then $x : T \in \Delta$ and for all $X \leq T : K \in \Gamma$ then $X \leq T : K \in \Delta$.

COROLLARY 3.21 (*Weakening/Permutation*) Let Γ and Γ' be contexts such that $\Gamma \subseteq \Delta$ and $\Delta \vdash \text{ok}$. Then $\Gamma \vdash \Sigma$ implies $\Delta \vdash \Sigma$.

PROOF: Let γ be the identity substitution. Then γ is a renaming for Γ in Δ and $\Sigma[\gamma] \equiv \Sigma$. Then, by Renaming (Proposition 3.19), it follows that $\Delta \vdash \Sigma$. \square

LEMMA 3.22 (*Context, kind, and term strengthening*)

1. If $\Gamma_1, X \leq T : K, \Gamma_2 \vdash \text{ok}$ and $X \notin \text{FTV}(\Gamma_2)$, then $\Gamma_1, \Gamma_2 \vdash \text{ok}$.
2. If $\Gamma_1, X \leq T : K, \Gamma_2 \vdash S : K'$ and $X \notin \text{FTV}(\Gamma_2) \cup \text{FTV}(S)$, then $\Gamma_1, \Gamma_2 \vdash S : K'$.
3. If $\Gamma_1, x : T, \Gamma_2 \vdash \Sigma$ and $x \notin \text{FV}(\Sigma)$, then $\Gamma_1, \Gamma_2 \vdash \Sigma$.

Moreover, the derivations of the conclusions are strictly shorter than the derivation of the premises.

PROOF: Statements 1 and 2 follow by simultaneous induction on the length of derivations, and statement 3 by induction on the derivation of $\Gamma_1, x : T, \Gamma_2 \vdash \Sigma$. In all cases lemmas 3.15 and 3.17 are used. \square

PROPOSITION 3.23 (*Generation for kinding*)

1. $\Gamma \vdash X : K$ implies $\Gamma \equiv \Gamma_1, X \leq T : K, \Gamma_2$ for some Γ_1, T , and Γ_2 .
2. $\Gamma \vdash T_1 \rightarrow T_2 : K$ implies $K \equiv \star$ and $\Gamma \vdash T_1, T_2 : \star$.
3. $\Gamma \vdash \forall X \leq T_1 : K_1. T_2 : K$ implies $K \equiv \star$ and $\Gamma, X \leq T_1 : K_1 \vdash T_2 : \star$.
4. $\Gamma \vdash \Lambda(X : K_1) T_2 : K$ implies $K \equiv K_1 \rightarrow K_2$ and $\Gamma, X \leq \top^{K_1} : K_1 \vdash T_2 : K_2$, for some K_2 .
5. $\Gamma \vdash S T : K$ implies $\Gamma \vdash S : K' \rightarrow K$ and $\Gamma \vdash T : K'$, for some K' .
6. $\Gamma \vdash \bigwedge^K [T_1 \dots T_n] : K'$ implies $K \equiv K'$ and $\Gamma \vdash \text{ok}$ and $\Gamma \vdash T_i : K$ for each i .

Moreover, the proofs of the consequents are all strictly shorter than those of the antecedents.

PROOF: In each case the antecedent uniquely determines the last rule of its derivation. The proof follows by inspection of the rules. \square

LEMMA 3.24 (*Uniqueness of kinds*) If $\Gamma \vdash T : K$ and $\Gamma \vdash T : K'$, then $K \equiv K'$.

LEMMA 3.25 (*Type substitution*) Let $\Gamma_1 \vdash T : K_U$. Then

1. If $\Gamma_1, X \leq U : K_U, \Gamma_2 \vdash S : K_S$, then $\Gamma_1, \Gamma_2[X \leftarrow T] \vdash S[X \leftarrow T] : K_S$.
2. If $\Gamma_1, X \leq U : K_U, \Gamma_2 \vdash \text{ok}$, then $\Gamma_1, \Gamma_2[X \leftarrow T] \vdash \text{ok}$.

PROOF: By simultaneous induction on derivations of the premises. The proof of part 2 is straightforward using part 1 of the induction hypothesis. We consider the details of the proof of 1. The cases K-ARROW, K-ALL, K-OABS, and K-OAPP follow by straightforward application of part 1 of the induction hypothesis and the corresponding rule, while the case of K-MEET also uses part 2 of the induction hypothesis. We examine the case of K-TVAR, where $S \equiv Y$ for some variable Y . By proposition 3.23(1) $Y \leq T_Y : K_S \in (\Gamma_1, X \leq U : K_U, \Gamma_2)$ for some T_Y . There are three cases to consider.

$Y \leq T_Y : K_S \in \Gamma_1$ Then we also have $Y \leq T_Y : K_S \in (\Gamma_1, \Gamma_2[X \leftarrow T])$. By part 2 of the induction hypothesis, $\Gamma_1, \Gamma_2[X \leftarrow T] \vdash \text{ok}$. Applying K-TVAR, we get $\Gamma_1, \Gamma_2[X \leftarrow T] \vdash Y : K_S$.

$Y \leq T_Y : K_S \equiv X \leq U : K_U$ We know that $\Gamma_1 \vdash T : K_S \equiv K_U$. From the premise of K-TVAR and part 2 of the induction hypothesis, we have $\Gamma_1, \Gamma_2[X \leftarrow T] \vdash \text{ok}$. The result follows by weakening (corollary 3.21).

$Y \leq T_Y : K_S \in \Gamma_2$ Then we have $Y \leq T_Y[X \leftarrow T] : K_S \in (\Gamma_1, \Gamma_2[X \leftarrow T])$. By part 2 of the induction hypothesis, $\Gamma_1, \Gamma_2[X \leftarrow T] \vdash \text{ok}$, from which the result follows by K-TVAR. \square

LEMMA 3.26 (*Subject reduction for kinding judgements*) If $S \rightarrow_{\beta\wedge} T$ and $\Gamma \vdash S : K$, then $\Gamma \vdash T : K$.

PROOF: In order to prove this result it is enough to prove the following statements by simultaneous induction on the derivation of $\Gamma \vdash S : K$. The rest follows by induction on the definition of $\rightarrow_{\beta\wedge}$.

1. $\Gamma \vdash \text{ok}$ and $\Gamma \rightarrow_{\beta\wedge} \Gamma'$ implies $\Gamma' \vdash \text{ok}$.
2. $\Gamma \vdash S : K$ and $S \rightarrow_{\beta\wedge} T$ implies $\Gamma \vdash T : K$.
3. $\Gamma \vdash S : K$ and $\Gamma \rightarrow_{\beta\wedge} \Gamma'$ implies $\Gamma' \vdash S : K$. \square

THEOREM 3.27 (*Kind invariance under type conversion*) If $\Gamma \vdash S : K_S$ and $\Gamma \vdash T : K_T$, with $S =_{\beta\wedge} T$, then $K_S \equiv K_T$.

PROOF: By the Church-Rosser theorem 3.5, there exists U such that $S \rightarrow_{\beta\wedge} U$ and $T \rightarrow_{\beta\wedge} U$, and the result follows by subject reduction and uniqueness of kinds. \square

LEMMA 3.28 Let $\Gamma \vdash S_i : K$ for every $i \in \{1..n\}$. If for every j in $\{1..m\}$ there exists i in $\{1..n\}$ such that $\Gamma \vdash S_i \leq T_j$, then $\Gamma \vdash \bigwedge^K [S_1..S_n] \leq \bigwedge^K [T_1..T_m]$.

A particular case of the previous lemma is the following.

COROLLARY 3.29 Let $\Gamma \vdash S_i : K$ for each $i \in \{1..n\}$. Then $\Gamma \vdash S_i \leq T_i$, for every $i \in \{1..n\}$, implies $\Gamma \vdash \bigwedge^K [S_1..S_n] \leq \bigwedge^K [T_1..T_n]$.

LEMMA 3.30 Let $\Gamma \vdash S, T : K$. Then $\Gamma \vdash S \leq T$ if and only if $\Gamma \vdash S^{nf} \leq T^{nf}$.

PROOF: We shall consider only one part, the other is similar.

\Rightarrow) By subject reduction, we have that $\Gamma \vdash S^{nf} : K$, then, by S-CONV, $\Gamma \vdash S^{nf} \leq S$. By similar reasoning we have $\Gamma \vdash T \leq T^{nf}$. The result follows by applying S-TRANS twice. \square

LEMMA 3.31 (*Context modification*) If $\Gamma_1 \vdash U' : K$ and Σ is either ok or $T : K'$, then $\Gamma_1, X \leq U : K, \Gamma_2 \vdash \Sigma$ implies $\Gamma_1, X \leq U' : K, \Gamma_2 \vdash \Sigma$.

PROPOSITION 3.32 (*Well-kindedness of subtyping*) If $\Gamma \vdash S \leq T$, then $\Gamma \vdash S : K$ and $\Gamma \vdash T : K$ for some K .

PROOF: By induction on the derivation of $\Gamma \vdash S \leq T$. We show a few representative cases.

S-CONV We are given that $\Gamma \vdash S : K$ and $\Gamma \vdash T : K'$ and $S =_{\beta} T$. By lemma 3.27, $K \equiv K'$.

S-TVAR We are given that $\Gamma_1, X \leq T : K, \Gamma_2 \vdash \text{ok}$. $\Gamma_1, X \leq T : K, \Gamma_2 \vdash X : K$ follows by K-TVAR. Moreover, by lemma 3.16, we have $\Gamma_1 \vdash T : K$, and by weakening (corollary 3.21), $\Gamma_1, X \leq T : K, \Gamma_2 \vdash T : K$.

S-ARROW We are given $\Gamma \vdash T_1 \leq S_1$ and $\Gamma \vdash S_2 \leq T_2$ and $\Gamma \vdash S_1 \rightarrow S_2 : \star$. By proposition 3.23, $\Gamma \vdash S_1, S_2 : \star$. Further, by the induction hypothesis together with uniqueness of kinds (lemma 3.24), we have $\Gamma \vdash T_1, T_2 : \star$. Finally, the result follows by applying K-ARROW. \square

PROPOSITION 3.33 (*Well-kindedness of typing*) If $\Gamma \vdash e : T$, then $\Gamma \vdash T : \star$.

PROOF: By induction on the derivation of $\Gamma \vdash e : T$. We show here a few interesting cases.

T-VAR We are given $\Gamma_1, x:T, \Gamma_2 \vdash \text{ok}$. The result follows by generation for context judgements (lemma 3.16) and weakening (corollary 3.21).

T-ABS We are given $\Gamma, x:T_1 \vdash e : T_2$. By the induction hypothesis, $\Gamma, x:T_1 \vdash T_2 : \star$. By lemma 3.22, it follows that $\Gamma \vdash T_2 : \star$. Furthermore, by lemmas 3.15 and 3.16, $\Gamma \vdash T_1 : \star$. Hence, K-ARROW yields $\Gamma \vdash T_1 \rightarrow T_2 : \star$.

T-TAPP We know that $\Gamma \vdash f : \forall(X \leq T_1 : K_1)T_2$ and also $\Gamma \vdash S \leq T_1$. By the induction hypothesis, $\Gamma \vdash \forall(X \leq T_1 : K_1)T_2 : \star$ and, by proposition 3.23, $\Gamma, X \leq T_1 : K_1 \vdash T_2 : \star$. By lemmas 3.15 and 3.16, there exists a derivation of $\Gamma \vdash T_1 : K_1$. By the well-kindedness of subtyping (proposition 3.32) and uniqueness of kinds (lemma 3.24), we have $\Gamma \vdash S : K_1$. Then, by the type substitution lemma (lemma 3.25), $\Gamma \vdash T_2[X \leftarrow S] : \star$.

T-SUB By the induction hypothesis, proposition 3.32 and lemma 3.24. \square

3.3 Strong normalization of $\rightarrow_{\beta\wedge}$

We prove that every type that has a kind in F_{\wedge}^{ω} is strongly normalizing in three steps. We first prove that \rightarrow_a and also $\rightarrow_{\beta\wedge-}$ are strongly normalizing. The proof of strong normalization of $\rightarrow_{\beta\wedge-}$ on well-kinded types follows Tait's computability method [52]. Then we prove that both reductions commute, i.e. if $T \rightarrow_a T_1$ and $T_1 \rightarrow_{\beta\wedge-} T_2$, then there exists S such that $S \rightarrow_a T_2$ and $T \rightarrow_{\beta\wedge-}^{>0} S$ (in at least one step). Finally, using the previous two steps we prove that $\rightarrow_{\beta\wedge}$ is strongly normalizing.

A type T is called *strongly normalizing* if and only if all reduction sequences starting with T terminate. We write \mathbb{T} for the set of all type expressions and SN for the subset of \mathbb{T} of strongly normalizing type expressions. If A and B are subsets of \mathbb{T} , then $A \rightarrow B$ denotes the following subset of \mathbb{T}

$$A \rightarrow B = \{F \subseteq \mathbb{T} \mid \text{for all } a \in A, Fa \in B\}.$$

LEMMA 3.34 \rightarrow_a is strongly normalizing.

PROOF: By induction on the number of intersection symbols of the type expression being reduced.
□

To prove strong normalization of $\rightarrow_{\beta\wedge}$ we use a model-theoretic argument interpreting kinds as sets of normalizing terms, and the soundness of the model gives, as a corollary, the strong normalization property. The interpretation of a kind K , notation $\llbracket K \rrbracket$, is defined as follows.

$$\begin{aligned} \llbracket \star \rrbracket &= SN \\ \llbracket K_1 \rightarrow K_2 \rrbracket &= \llbracket K_1 \rrbracket \rightarrow \llbracket K_2 \rrbracket. \end{aligned}$$

DEFINITION 3.35 (*Saturated set*) $\mathcal{S} \subseteq SN$ is *saturated* if it satisfies the following conditions:

1. If $R_1..R_n \in SN$, then $XR_1..R_n \in \mathcal{S}$.
2. If $R_1..R_n, Q \in SN$, then
 - (a) if $P[X \leftarrow Q]R_1..R_n \in \mathcal{S}$, then $(\wedge X:K.P)QR_1..R_n \in \mathcal{S}$, for every K and
 - (b) if $(\wedge^{K_2}[T_1Q, \dots, T_mQ])R_1, \dots, R_n \in \mathcal{S}$,
then $(\wedge^{K_1 \rightarrow K_2}[T_1, \dots, T_m])QR_1, \dots, R_n \in \mathcal{S}$, for every K_1 .

Intuitively, a set of strongly normalizing type expressions is saturated if it contains all type variables and is closed under expansion of expressions which may have a kind of the form $K_1 \rightarrow K_2$.

LEMMA 3.36

1. SN is saturated.
2. If A, B are saturated, then $A \rightarrow B$ is saturated.
3. For any kind K , $\llbracket K \rrbracket$ is saturated.

DEFINITION 3.37

1. A valuation ρ in \mathbb{T} is a mapping from type variables to types.
2. The interpretation of a type with respect to ρ is

$$\llbracket T \rrbracket_\rho = T[X_1 \leftarrow \rho(X_1)..X_n \leftarrow \rho(X_n)],$$

where $FV(T) = \{X_1..X_n\}$.

3. Let ρ be a valuation in \mathbb{T} . Then ρ *satisfies* $T : K$, written $\rho \models T : K$, if $\llbracket T \rrbracket_\rho : \llbracket K \rrbracket$ and ρ *satisfies* $X \leq T : K$, written $\rho \models X \leq T : K$, if $\rho(X) : \llbracket K \rrbracket$. We say that ρ *satisfies* a context Γ , $\rho \models \Gamma$, if $\rho \models X \leq S : K$ for all $X \leq S : K : \Gamma$.
4. A context Γ *satisfies* $T : K$, written $\Gamma \models T : K$, if for every ρ such that $\rho \models \Gamma$, it follows that $\rho \models T : K$.

LEMMA 3.38

1. $\top^K \in \llbracket K \rrbracket$.
2. If $A_i \in \llbracket K \rrbracket$ for each $i \in \{1..n\}$, then $\wedge^K[A_1..A_n] \in \llbracket K \rrbracket$.

PROOF: We show item 2. Item 1 also follows follows by induction on the structure of K .

$K \equiv \star$ Then, by definition of $\llbracket K \rrbracket$, $A_i \in SN$ for each $i \in \{1..n\}$. Since every reduction starting from $\wedge^K[A_1..A_n]$ is a reduction consisting only of steps inside the A_i 's, one has $\wedge^K[A_1..A_n] \in SN \equiv \llbracket K \rrbracket$.

$K \equiv K_1 \rightarrow K_2$ Let $B \in \llbracket K_1 \rrbracket$. By the definition of \rightarrow , $A_i B \in \llbracket K_2 \rrbracket$, for each $i \in \{1..n\}$. By the induction hypothesis, $\bigwedge^{K_2} [A_1 B .. A_n B] \in \llbracket K_2 \rrbracket$. Moreover, $\bigwedge^{K_1 \rightarrow K_2} [A_1 .. A_n] B \in \llbracket K_2 \rrbracket$ by the saturation of $\llbracket K_2 \rrbracket$, which means that $\bigwedge^{K_1 \rightarrow K_2} [A_1 .. A_n] \in \llbracket K_1 \rightarrow K_2 \rrbracket$. \square

PROPOSITION 3.39 (*Soundness*) If $\Gamma \vdash T : K$, then $\Gamma \models T : K$.

PROOF: By induction on the derivation of $\Gamma \vdash T : K$.

We consider the case for K-MEET. The other cases follow by similar reasoning. Let $T \equiv \bigwedge^K [T_1 .. T_n]$. We have to consider two cases.

$n \neq 0$ We are given $\Gamma \vdash T_i : K$ for each $i \in \{1..n\}$, and, by the induction hypothesis, $\Gamma \models T_i : K$.

Let ρ be a valuation such that $\rho \models \Gamma$. Then $\llbracket T_i \rrbracket_\rho \in \llbracket K \rrbracket$, for each $i \in \{1..n\}$. By lemma 3.38(2), $\bigwedge^K [\llbracket T_1 \rrbracket_\rho .. \llbracket T_n \rrbracket_\rho] \in \llbracket K \rrbracket$.

$n \equiv 0$ $T \equiv \top^K$. Since $\llbracket \top^K \rrbracket_\rho \equiv \top^K$, the result follows by 3.38(1). \square

THEOREM 3.40 (*Strong normalization for $\rightarrow_{\beta\wedge^-}$*)

$\Gamma \vdash T : K$ implies that every $(\beta\wedge^-)$ -reduction sequence starting from T is finite.

PROOF: By soundness, $\Gamma \models T : K$. Choose ρ_0 such that $\rho_0(X) = X$. Observe that $\rho_0 \models \Gamma$ trivially. Hence $T \equiv \llbracket T \rrbracket_{\rho_0} \in \llbracket K \rrbracket \subseteq SN$. \square

LEMMA 3.41 If $T \rightarrow_a T_1$ and $T_1 \rightarrow_{\beta\wedge^-} T_2$, then there exists S such that $T \rightarrow_{\beta\wedge^-}^{>0} S$ and $S \rightarrow_a T_2$.

PROOF: By induction on the structure of T . \square

COROLLARY 3.42 (*\rightarrow_a postponement*) If $T \rightarrow_a T_1$ and $T_1 \rightarrow_{\beta\wedge^-} T_2$, then there exists S such that $T \rightarrow_{\beta\wedge^-}^{>0} S$ and $S \rightarrow_a T_2$.

PROOF: By induction on the generation of $T \rightarrow_a T_1$. \square

Finally, we can prove strong normalization for $\rightarrow_{\beta\wedge}$.

THEOREM 3.43 (*Strong normalization for $\rightarrow_{\beta\wedge}$*) $\Gamma \vdash T : K$ implies that every $(\beta\wedge)$ -reduction sequence starting from T is finite.

PROOF: Let $\Gamma \vdash T : K$. We reason by contradiction. Assume that there is an infinite $\beta\wedge$ -reduction sequence starting from T . Then lemma 3.34 and theorem 3.40 imply that there are infinitely many alternations of \rightarrow_a and $\rightarrow_{\beta\wedge^-}$ reduction sequences. By corollary 3.42, we can construct an infinite $(\beta\wedge^-)$ -reduction which contradicts theorem 3.40. \square

3.4 Towards a generation principle for subtyping

In this section we start our quest towards a generation principle for the subtyping relation of F_\wedge^ω . First, we develop a *normal subtyping system*, NF_\wedge^ω , in which only types in normal form are considered. We then prove that proofs in NF_\wedge^ω can be normalized by eliminating transitivity and simplifying reflexivity. This simplification yields an algorithmic presentation, $AlgF_\wedge^\omega$, whose rules are syntax directed. Moreover, we prove that $AlgF_\wedge^\omega$ is indeed an alternative presentation of the F_\wedge^ω subtyping relation. Formally, $\Gamma \vdash S \leq T$ if and only if $\Gamma^{nf} \vdash_{Alg} S^{nf} \leq T^{nf}$, when S and T are well-formed (proposition 3.74).

In the solution for the second order lambda calculus presented in [47], the distributivity rules for intersection types are not considered as rewrite rules. For that reason, new syntactic categories have to be defined (composite and individual canonical types) and an auxiliary mapping (flattening) transforms a type into a canonical type. Our solution does not need either new syntactic categories or elaborate auxiliary mappings, since the role played there by canonical types is performed here by types in normal form.

3.4.1 Normal subtyping

An important property of derivation systems is the information that a derivable judgement contains about its proofs. This information is essential to produce results which not only state properties about the subproofs, but also help identify ill-formed judgements.

EXAMPLE 3.44 As we mentioned in the introduction, in F_λ^ω we can prove:

$$W:K, X \leq \Lambda Y:K.Y:K \rightarrow K, Z \leq X:K \rightarrow K \vdash X(ZW) \leq W$$

Note that X and Z are subtypes of the identity on K , therefore it makes sense for $X(ZW)$ to be a subtype of W . The derivation is as follows: Let $\Gamma \equiv W:K, X \leq \Lambda Y:K.Y:K \rightarrow K, Z \leq X:K \rightarrow K$. For the sake of readability we omit kinding judgements.

$$\frac{\frac{\frac{\Gamma \vdash \text{ok}}{\Gamma \vdash X \leq \Lambda Y:K.Y} \text{S-TVAR} \quad \frac{(\Lambda Y:K.Y)(ZW) =_{\beta\wedge} ZW}{(\Lambda Y:K.Y)(ZW) \leq ZW} \text{S-CONV}}{\Gamma \vdash X(ZW) \leq (\Lambda Y:K.Y)(ZW)} \text{S-OAPP}}{\Gamma \vdash X(ZW) \leq ZW} \text{S-TRANS}$$

$$\frac{\frac{\frac{\Gamma \vdash \text{ok}}{\Gamma \vdash Z \leq X} \text{S-TVAR} \quad \frac{\Gamma \vdash \text{ok}}{\Gamma \vdash X \leq \Lambda Y:K.Y} \text{S-TVAR}}{\Gamma \vdash Z \leq (\Lambda Y:K.Y)} \text{S-TRANS} \quad \frac{(\Lambda Y:K.Y)W =_{\beta\wedge} W}{(\Lambda Y:K.Y)W \leq W} \text{S-CONV}}{\Gamma \vdash ZW \leq (\Lambda Y:K.Y)W} \text{S-OAPP}}{\Gamma \vdash ZW \leq W} \text{S-TRANS}$$

$$\frac{\Gamma \vdash X(ZW) \leq ZW \quad \Gamma \vdash ZW \leq W}{\Gamma \vdash X(ZW) \leq W} \text{S-TRANS}$$

This simple example already shows that S-TRANS erases information obtained by S-CONV that is not present in the conclusion any longer. A first step towards an algorithm to check the subtyping relation is to design a set of rules in which the derivable judgements contain all the information about their derivations. To this end we define a set of rules, NF_λ^ω , in which conversion is reduced to a minimum and, as we show in lemmas 3.53 and 3.54, reflexivity can be simplified and transitivity can be eliminated. Both results are proved with a standard cut-elimination argument. This yields a syntax directed subtyping relation, $AlgF_\lambda^\omega$, which constitutes a decision procedure for the original system. In section 4.2.1 we prove the termination of $AlgF_\lambda^\omega$.

In the rest of this section, we present the subtyping system NF_λ^ω , which uses the context and type formation rules of F_λ^ω . We define rewriting rules for derivations in NF_λ^ω (definitions 3.51 and 3.52), and describe a terminating procedure to normalize proofs, which gives, as a consequence, the generation for subtyping (proposition 3.58) and an algorithmic presentation, $AlgF_\lambda^\omega$ (see section 3.7). Finally, in section 3.7, we show that there is an equivalence between subtyping in F_λ^ω and subtyping in $AlgF_\lambda^\omega$.

We now define the *normal subtyping system*, NF_λ^ω . Subtyping judgements in NF_λ^ω are written $\Gamma \vdash_n S \leq T$, and S, T , and all types appearing in Γ are in $\beta\wedge$ -normal form.

NOTATION 3.45 A, B , and C range over types whose outermost constructor is not an intersection.

REMARK 3.46 It is an immediate consequence of the $\beta\wedge$ -reduction rules that, if T is in $\beta\wedge$ -normal form, then T is either $X, S \rightarrow A, \forall X \leq S:K.A, \Lambda X:K.A$, or AS where A is not an abstraction, or $\bigwedge^K[A_1..A_n]$. We frequently use this notation as a reminder of the shape of types in normal form.

We now define $\text{lub}_\Gamma(S)$. We prove in lemma 3.60 and corollary 3.70, that, when defined, it is the smallest type with respect to the subtyping order beyond S with respect to the subtyping assumption in Γ . In other words, there is no type between X and $\Gamma(X)$ (modulo $\beta\wedge$ -equality). If $\Gamma \vdash X \leq T$ then $\Gamma \vdash \Gamma(X) \leq T$.

DEFINITION 3.47 (*Least strict Upper Bound*)

$$\begin{aligned} \text{lub}_\Gamma(X) &= \Gamma(X), \\ \text{lub}_\Gamma(T S) &= \text{lub}_\Gamma(T) S. \end{aligned}$$

DEFINITION 3.48 (NF_\wedge^ω *subtyping rules*)

$$\begin{aligned} &\frac{\Gamma \vdash S : K}{\Gamma \vdash_n S \leq S} && \text{(NS-REFL)} \\ &\frac{\Gamma \vdash_n S \leq T \quad \Gamma \vdash_n T \leq U}{\Gamma \vdash_n S \leq U} && \text{(NS-TRANS)} \\ &\frac{\Gamma \vdash_n \Gamma(X) \leq A \quad X \neq A}{\Gamma \vdash_n X \leq A} && \text{(NS-TVAR)} \\ &\frac{\Gamma \vdash_n T \leq S \quad \Gamma \vdash_n A \leq B \quad \Gamma \vdash S \rightarrow A : \star}{\Gamma \vdash_n S \rightarrow A \leq T \rightarrow B} && \text{(NS-ARROW)} \\ &\frac{\Gamma, X \leq S : K \vdash_n A \leq B \quad \Gamma \vdash \forall X \leq S : K. A : \star}{\Gamma \vdash_n \forall X \leq S : K. A \leq \forall X \leq S : K. B} && \text{(NS-ALL)} \\ &\frac{\Gamma, X \leq \top^K : K \vdash_n A \leq B}{\Gamma \vdash_n \wedge X : K. A \leq \wedge X : K. B} && \text{(NS-OABS)} \\ &\frac{\Gamma \vdash_n (\text{lub}_\Gamma(A S))^{nf} \leq B \quad \Gamma \vdash A S : K \quad A S \neq B}{\Gamma \vdash_n A S \leq B} && \text{(NS-OAPP)} \\ &\frac{\forall i \in \{1..m\} \Gamma \vdash_n A \leq B_i \quad \Gamma \vdash A : K}{\Gamma \vdash_n A \leq \bigwedge^K [B_1..B_m]} && \text{(NS-V)} \\ &\frac{\exists j \in \{1..n\} \Gamma \vdash_n A_j \leq A \quad \forall k \in \{1..n\} \Gamma \vdash A_k : K}{\Gamma \vdash_n \bigwedge^K [A_1..A_n] \leq A} && \text{(NS-}\exists\text{)} \\ &\frac{\forall i \in \{1..m\} \exists j \in \{1..n\} \Gamma \vdash_n A_j \leq B_i \quad \forall k \in \{1..n\} \Gamma \vdash A_k : K}{\Gamma \vdash_n \bigwedge^K [A_1..A_n] \leq \bigwedge^K [B_1..B_m]} && \text{(NS-}\forall\exists\text{)} \end{aligned}$$

As we mentioned in the introduction, an important factor to develop this system was to consider the distributivity rules of the presentation of F_\wedge^ω in [30] as reduction rules instead of subtyping rules. This new point of view suggested that an algorithmic system should, to a certain extent, concentrate on normal forms replacing the conversion rule by reflexivity. Consequently, a derivation of a subtyping statement should involve only types in normal form. But enlightened by the simple (counter)example 3.44 it is not possible to perform all reductions at once. In other words, the system does not satisfy an S-CONV postponement property. Without using S-CONV it is not possible to derive example 3.44. Hence, the solution is not as simple as replacing S-CONV by NS-REFL.

In general, the interaction between S-TRANS and S-CONV can be analyzed as follows. In S-TRANS the metavariable T of the hypothesis is not present in the conclusion, but this is not a problem by itself (a similar situation appears in the simply typed lambda calculus in its application rule and the system is deterministic). The problem is that in the presence of S-CONV the *vanishing* T can be $\beta\wedge$ -convertible to either S or U or to both S and U . What example 3.44 shows is that S and U may be different normal forms, which means that searching for T is inherently nondeterministic.

We cannot eliminate transitivity completely, we still need it on type variables and on type applications. In F_{\leq} [37] transitivity is eliminated and hidden in a richer variable rule in which deciding whether $\Gamma \vdash X \leq T$ when $T \neq X$ is reduced to deciding whether the bound of X is smaller than or equal to T . The bound of X has the particular property of being the *least strict upper bound* of X . This observation motivated the definition of our NS-OAPP rule, in which we reduce the decision of whether $\Gamma \vdash A S \leq B$ when $B \neq A S$, to checking if the least strict upper bound

of AS is smaller than or equal to B (See lemma 3.60 and corollary 3.70). The least strict upper bound of AS , $\text{lub}_\Gamma(AS)$, is obtained from AS by replacing its leftmost innermost variable by the corresponding bound in Γ . In our example, $\text{lub}_\Gamma(X(ZW))$ is $(\Lambda Y:K.Y)(ZW)$. Consequently, $\text{lub}_\Gamma(AS)$ may be other than a normal form. That is the reason we normalize it. The strength of the conversion rule that is not captured by reflexivity is hidden in this normalization step. Since AS is a well kinded type, by the free variables lemma (lemma 3.17), $\text{FTV}(AS) \subseteq \text{dom}(\Gamma)$. Therefore, $\text{lub}_\Gamma(AS)$ is defined. By lemma 3.60(1), $\text{lub}_\Gamma(AS)$ is well-kinded, and since well-kinded types are strongly normalizing, its normal form exists. The rules S-MEET-LB and S-MEET-G are replaced by NS- \exists , NS- \forall , and NS- $\forall\exists$. Except for the restriction of types being in normal form NS-ARROW, NS-ALL, and NS-OABS have the same form as S-ARROW, S-ALL, and S-OABS respectively.

3.5 Structural properties of NF_\wedge^ω

This section establishes a number of structural properties of NF_\wedge^ω . The proofs of lemmas 3.49 and 3.50 are similar to those of the corresponding properties for F_\wedge^ω .

LEMMA 3.49 If $\Gamma \vdash_n S \leq T$ and Γ_1 is a prefix of Γ , then $\Gamma_1 \vdash \text{ok}$ as a subderivation. Moreover, the subderivation is strictly shorter.

LEMMA 3.50 (*Weakening/Permutation*) Let Γ and Γ' be contexts such that $\Gamma \subseteq \Gamma'$ and $\Gamma' \vdash \text{ok}$. Then $\Gamma \vdash_n S \leq T$ implies $\Gamma' \vdash_n S \leq T$.

We present rewriting rules on derivations to simplify instances of NS-REFL and NS-TRANS. We give a terminating strategy to transform a given derivation into a derivation with occurrences of NS-REFL only applied to type variables or type applications and without occurrences of NS-TRANS. To improve readability we omit kinding judgements in the transitivity elimination rules which appear as hypotheses in the redex or in a proper subderivation of the missing ones, as we proved in generation for kinding (proposition 3.23). The derivations of the kinding judgements of each reduct of the reflexivity rules are proper subderivations of the kinding judgements in its redex.

DEFINITION 3.51 (*Reflexivity simplification rules*)

$$\begin{array}{l}
1. \quad \boxed{\frac{\Gamma \vdash S \rightarrow A : \star}{\Gamma \vdash_n S \rightarrow A \leq S \rightarrow A} \text{NS-REFL}} \\
\Rightarrow_R \quad \boxed{\frac{\frac{\Gamma \vdash S : \star}{\Gamma \vdash_n S \leq S} \text{NS-REFL} \quad \frac{\Gamma \vdash A : \star}{\Gamma \vdash_n A \leq A} \text{NS-REFL}}{\Gamma \vdash_n S \rightarrow A \leq S \rightarrow A} \text{NS-ARROW}} \\
2. \quad \boxed{\frac{\Gamma \vdash \forall X \leq S : K.A : \star}{\Gamma \vdash_n \forall X \leq S : K.A \leq \forall X \leq S : K.A} \text{NS-REFL}} \\
\Rightarrow_R \quad \boxed{\frac{\frac{\Gamma, X \leq S : K \vdash A : \star}{\Gamma, X \leq S : K \vdash_n A \leq A} \text{NS-REFL}}{\Gamma \vdash_n \forall X \leq S : K.A \leq \forall X \leq S : K.A} \text{NS-ALL}} \\
3. \quad \boxed{\frac{\Gamma \vdash \Lambda X : K.A : K \rightarrow K'}{\Gamma \vdash_n \Lambda X : K.A \leq \Lambda X : K.A} \text{NS-REFL}} \\
\Rightarrow_R \quad \boxed{\frac{\frac{\Gamma, X : K \vdash A : K'}{\Gamma, X : K \vdash_n A \leq A} \text{NS-REFL}}{\Gamma \vdash_n \Lambda X : K.A \leq \Lambda X : K.A} \text{NS-OABS}}
\end{array}$$

$$4. \quad \boxed{\frac{\Gamma \vdash \bigwedge^K [A_1 \dots A_n] : K}{\Gamma \vdash_n \bigwedge^K [A_1 \dots A_n] \leq \bigwedge^K [A_1 \dots A_n]} \text{NS-REFL}}$$

$$\Rightarrow_R \boxed{\frac{\frac{\Gamma \vdash A_i : K}{\Gamma \vdash_n A_i \leq A_i \forall i \in \{1..n\}} \text{NS-REFL}}{\Gamma \vdash_n \bigwedge^K [A_1 \dots A_n] \leq \bigwedge^K [A_1 \dots A_n]} \text{NS-}\forall\exists}$$

DEFINITION 3.52 (*Transitivity elimination rules*)

$$1. \quad \boxed{\frac{\frac{\Gamma \vdash S : K}{\Gamma \vdash_n S \leq S} \text{NS-REFL} \quad \Gamma \vdash_n S \leq T}{\Gamma \vdash_n S \leq T} \text{NS-TRANS}} \Rightarrow_T \boxed{\Gamma \vdash_n S \leq T}$$

$$2. \quad \boxed{\frac{\Gamma \vdash T : K}{\Gamma \vdash_n S \leq T \quad \Gamma \vdash_n T \leq T} \text{NS-REFL}} \Rightarrow_T \boxed{\Gamma \vdash_n S \leq T}$$

$$\frac{\Gamma \vdash_n S \leq T \quad \Gamma \vdash_n T \leq T}{\Gamma \vdash_n S \leq T} \text{NS-TRANS}$$

$$3. \quad \boxed{\frac{\Gamma \vdash_n \Gamma(X) \leq A}{\Gamma \vdash_n X \leq A} \text{NS-TVAR} \quad \Gamma \vdash_n A \leq B}{\Gamma \vdash_n X \leq B} \text{NS-TRANS}}$$

$$\Rightarrow_T \boxed{\frac{\frac{\Gamma \vdash_n \Gamma(X) \leq A \quad \Gamma \vdash_n A \leq B}{\Gamma \vdash_n \Gamma(X) \leq B} \text{NS-TRANS}}{\Gamma \vdash_n X \leq B} \text{NS-TVAR}}$$

$$4. \quad \boxed{\frac{\frac{\Gamma \vdash_n T \leq S \quad \Gamma \vdash_n A \leq B}{\Gamma \vdash_n S \rightarrow A \leq T \rightarrow B} \text{NS-ARROW} \quad \frac{\Gamma \vdash_n U \leq T \quad \Gamma \vdash_n B \leq C}{\Gamma \vdash_n T \rightarrow B \leq U \rightarrow C} \text{NS-ARROW}}{\Gamma \vdash_n S \rightarrow A \leq U \rightarrow C} \text{NS-TRANS}}$$

$$\Rightarrow_T \boxed{\frac{\frac{\Gamma \vdash_n U \leq T \quad \Gamma \vdash_n T \leq S}{\Gamma \vdash_n U \leq S} \text{NS-TRANS} \quad \frac{\Gamma \vdash_n A \leq B \quad \Gamma \vdash_n B \leq C}{\Gamma \vdash_n A \leq C} \text{NS-TRANS}}{\Gamma \vdash_n S \rightarrow A \leq U \rightarrow C} \text{NS-ARROW}}$$

$$5. \quad \boxed{\frac{\frac{\Gamma, X \leq S : K \vdash_n A \leq B}{\Gamma \vdash_n \forall X \leq S : K. A \leq \forall X \leq S : K. B} \text{NS-ALL} \quad \frac{\Gamma, X \leq S : K \vdash_n B \leq C}{\Gamma \vdash_n \forall X \leq S : K. B \leq \forall X \leq S : K. C} \text{NS-ALL}}{\Gamma \vdash_n \forall X \leq S : K. A \leq \forall X \leq S : K. C} \text{NS-TRANS}}$$

$$\Rightarrow_T \boxed{\frac{\frac{\Gamma, X \leq S : K \vdash_n A \leq B \quad \Gamma, X \leq S : K \vdash_n B \leq C}{\Gamma, X \leq S : K \vdash_n A \leq C} \text{NS-TRANS}}{\Gamma \vdash_n \forall X \leq S : K. A \leq \forall X \leq S : K. C} \text{NS-ALL}}$$

$$6. \quad \boxed{\frac{\frac{\Gamma, X : K \vdash_n A \leq B}{\Gamma \vdash_n \lambda X : K. A \leq \lambda X : K. B} \text{NS-OABS} \quad \frac{\Gamma, X : K \vdash_n B \leq C}{\Gamma \vdash_n \lambda X : K. B \leq \lambda X : K. C} \text{NS-OABS}}{\Gamma \vdash_n \lambda X : K. A \leq \lambda X : K. C} \text{NS-TRANS}}$$

$$\Rightarrow_T \frac{\frac{\Gamma, X:K \vdash_n A \leq B \quad \Gamma, X:K \vdash_n B \leq C}{\Gamma, X:K \vdash_n A \leq C} \text{NS-TRANS}}{\Gamma \vdash_n \Lambda X:K. A \leq \Lambda X:K. C} \text{NS-OABS}$$

$$7. \frac{\frac{\Gamma \vdash_n \text{lub}_\Gamma(A S)^{nf} \leq B}{\Gamma \vdash_n A S \leq B} \text{NS-OAPP} \quad \Gamma \vdash_n B \leq C}{\Gamma \vdash_n A S \leq C} \text{NS-TRANS}$$

$$\Rightarrow_T \frac{\frac{\Gamma \vdash_n (\text{lub}_\Gamma(A S))^{nf} \leq B \quad \Gamma \vdash_n B \leq C}{\Gamma \vdash_n \text{lub}_\Gamma(A S)^{nf} \leq C} \text{NS-TRANS}}{\Gamma \vdash_n A S \leq C} \text{NS-OAPP}$$

$$8. \frac{\frac{\forall i \in \{1..n\} \Gamma \vdash_n A \leq A_i}{\Gamma \vdash_n A \leq \bigwedge^K [A_1..A_n]} \text{NS-}\forall \quad \frac{\exists j \in \{1..n\} \Gamma \vdash_n A_j \leq B}{\Gamma \vdash_n \bigwedge^K [A_1..A_n] \leq B} \text{NS-}\exists}{\Gamma \vdash_n A \leq B} \text{NS-TRANS}$$

$$\Rightarrow_T \frac{\exists j \in \{1..n\} \Gamma \vdash_n A \leq A_j \quad \Gamma \vdash_n A_j \leq B}{\Gamma \vdash_n A \leq B} \text{NS-TRANS}$$

$$9. \frac{\frac{\forall i \in \{1..n\} \Gamma \vdash_n B \leq A_i}{\Gamma \vdash_n B \leq \bigwedge^K [A_1..A_n]} \text{NS-}\forall \quad \Gamma \vdash_n A \leq B}{\Gamma \vdash_n A \leq \bigwedge^K [A_1..A_n]} \text{NS-TRANS}$$

$$\Rightarrow_T \frac{\frac{\forall i \in \{1..n\} \Gamma \vdash_n A \leq B \quad \Gamma \vdash_n B \leq A_i}{\forall i \in \{1..n\} \Gamma \vdash_n A \leq A_i} \text{NS-TRANS}}{\Gamma \vdash_n A \leq \bigwedge^K [A_1..A_n]} \text{NS-}\forall$$

$$10. \frac{\frac{\exists j \in \{1..n\} \Gamma \vdash_n A_j \leq B}{\Gamma \vdash_n \bigwedge^K [A_1..A_n] \leq B} \text{NS-}\exists \quad \Gamma \vdash_n B \leq A}{\Gamma \vdash_n \bigwedge^K [A_1..A_n] \leq A} \text{NS-TRANS}$$

$$\Rightarrow_T \frac{\frac{\exists j \in \{1..n\} \Gamma \vdash_n A_j \leq B \quad \Gamma \vdash_n B \leq A}{\exists j \in \{1..n\} \Gamma \vdash_n A_j \leq A} \text{NS-TRANS}}{\Gamma \vdash_n \bigwedge^K [A_1..A_n] \leq A} \text{NS-}\exists$$

$$11. \frac{\frac{\exists j \in \{1..m\} \Gamma \vdash_n A_j \leq A}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq A} \text{NS-}\exists \quad \frac{\forall i \in \{1..n\} \Gamma \vdash_n A \leq B_i}{\Gamma \vdash_n A \leq \bigwedge^K [B_1..B_n]} \text{NS-}\forall}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq \bigwedge^K [B_1..B_n]} \text{NS-TRANS}$$

$$\Rightarrow_T \frac{\frac{\exists j \in \{1..m\} \Gamma \vdash_n A_j \leq A \quad \forall i \in \{1..n\} \Gamma \vdash_n A \leq B_i}{\forall i \in \{1..n\} \exists j \in \{1..m\} \Gamma \vdash_n A_j \leq B_i} \text{NS-TRANS}}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq \bigwedge^K [B_1..B_n]} \text{NS-}\forall\exists$$

$$\begin{array}{c}
12. \quad \frac{\frac{\frac{\forall i \in \{1..n\} \exists j \in \{1..m\} \Gamma \vdash_n A_j \leq B_i \quad \forall k \in \{1..r\} \exists i \in \{1..n\} \Gamma \vdash_n B_i \leq C_k}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq \bigwedge^K [B_1..B_n]} \text{NS-}\forall\exists}{\Gamma \vdash_n \bigwedge^K [B_1..B_n] \leq \bigwedge^K [C_1..C_r]} \text{NS-TRANS}}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq \bigwedge^K [C_1..C_r]} \text{NS-TRANS} \\
\Rightarrow_T \quad \frac{\frac{\frac{\forall k \in \{1..r\} \exists i \in \{1..n\} \exists j \in \{1..m\} \quad \Gamma \vdash_n A_j \leq B_i \quad \Gamma \vdash_n B_i \leq C_k}{\forall k \in \{1..r\} \exists j \in \{1..m\} \Gamma \vdash_n A_j \leq C_k} \text{NS-TRANS}}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq \bigwedge^K [C_1..C_r]} \text{NS-}\forall\exists} \\
13. \quad \frac{\frac{\frac{\forall i \in \{1..n\} \exists j \in \{1..m\} \Gamma \vdash_n A_j \leq B_i}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq \bigwedge^K [B_1..B_n]} \text{NS-}\forall\exists} \quad \frac{\frac{\exists i \in \{1..n\} \Gamma \vdash_n B_i \leq C}{\Gamma \vdash_n \bigwedge^K [B_1..B_n] \leq C} \text{NS-}\exists}}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq C} \text{NS-TRANS} \\
\Rightarrow_T \quad \frac{\frac{\frac{\exists j \in \{1..m\} \exists i \in \{1..n\} \Gamma \vdash_n A_j \leq B_i \quad \Gamma \vdash_n B_i \leq C}{\exists j \in \{1..m\} \Gamma \vdash_n A_j \leq C} \text{NS-TRANS}}{\Gamma \vdash_n \bigwedge^K [A_1..A_m] \leq C} \text{NS-}\exists} \\
14. \quad \frac{\frac{\frac{\forall i \in \{1..n\} \Gamma \vdash_n A \leq B_i}{\Gamma \vdash_n A \leq \bigwedge^K [B_1..B_n]} \text{NS-}\forall} \quad \frac{\frac{\forall k \in \{1..r\} \exists i \in \{1..n\} \Gamma \vdash_n B_i \leq C_k}{\Gamma \vdash_n \bigwedge^K [B_1..B_n] \leq \bigwedge^K [C_1..C_r]} \text{NS-}\forall\exists}}{\Gamma \vdash_n A \leq \bigwedge^K [C_1..C_r]} \text{NS-TRANS} \\
\Rightarrow_T \quad \frac{\frac{\frac{\forall k \in \{1..r\} \exists i \in \{1..n\} \quad \Gamma \vdash_n A \leq B_i \quad \Gamma \vdash_n B_i \leq C_k}{\forall k \in \{1..r\} \Gamma \vdash_n A \leq C_k} \text{NS-TRANS}}{\Gamma \vdash_n A \leq \bigwedge^K [C_1..C_r]} \text{NS-}\forall}
\end{array}$$

A derivation of a subtyping judgement is in *refl-normal form* if it has no reflexivity redexes and it is in *trans-normal form* if it has no transitivity redexes, and it is in *normal form* if it has neither reflexivity nor transitivity redexes. The elimination of NS-TRANS, and the simplification of NS-REFL follow a standard cut-elimination argument.

LEMMA 3.53 (*Reflexivity simplification*) Let D be a derivation of a subtyping judgement with only one application of NS-REFL. Then D has a refl-normal form.

PROOF: Same argument as in lemma 3.54. \square

LEMMA 3.54 (*Transitivity elimination*) Let D be a derivation of a subtyping judgement with only one application of NS-TRANS. Then D has a trans-normal form.

PROOF: By induction on the size of D following a case analysis of the last rule of D . If the last rule is not NS-TRANS, then the result follows by the induction hypothesis. Otherwise we consider all possible last rules of the derivations of the premises and note that each possible configuration determines a trans-redex. Finally, observe that each reduction yields either a derivation in normal form or shorter derivations with only one occurrence of NS-TRANS in which case the result follows by the induction hypothesis. \square

An immediate corollary of this last result is that transitivity elimination terminates. Given a derivation D of $\Gamma \vdash_n S \leq T$, iterate the previous lemma on all subderivations of D that have only one NS-TRANS application. The number of times the lemma is applied is equal to the number of occurrences of NS-TRANS in D . Furthermore, lemma 3.53 implies that reflexivity simplification terminates. The simplification rules are such that transitivity simplification rules do not create new reflexivity redexes. Therefore, we can reduce all instances of NS-REFL first and then all instances of NS-TRANS, which is a terminating procedure to normalize a derivation. Consequently, we have proved the following corollary.

COROLLARY 3.55 (*Existence of normal derivations*) Given a derivation of $\Gamma \vdash_n S \leq T$. Then there exists a derivation in normal form of $\Gamma \vdash_n S \leq T$.

LEMMA 3.56

1. A derivation in normal form whose last rule is NS-REFL is either a proof of $\Gamma \vdash_n X \leq X$ or of $\Gamma \vdash_n AT \leq AT$.
2. If the last rule of a subtyping derivation D is NS-TRANS, then D is not in normal form.

PROOF:

1. According to the reflexivity elimination rules, any other possible NS-REFL application is a redex.
2. By case analysis of the last rules of the premises of the last rule of D . In each case the result follows either by the induction hypothesis or because the last rule of at least one of the derivations of the premises of D constitutes a redex. \square

We can summarize the previous results as follows.

COROLLARY 3.57 If $\Gamma \vdash_n S \leq T$, then there exists a proof of the same judgement with no applications of NS-TRANS and in which NS-REFL is only applied to type variables and type applications.

A consequence of the normalization of proofs is the following generation result.

PROPOSITION 3.58 (*Generation for normal subtyping*)

1. $\Gamma \vdash_n X \leq B$ implies $X \equiv B$ and $\Gamma \vdash X : K$ for some K , or $\Gamma \vdash_n \Gamma(X) \leq B$.
2. $\Gamma \vdash_n S \rightarrow A \leq B$ implies $B \equiv T \rightarrow C$, $\Gamma \vdash_n T \leq S$, $\Gamma \vdash_n A \leq C$, and $\Gamma \vdash S \rightarrow A : \star$.
3. $\Gamma \vdash_n \forall X \leq S : K. A \leq B$ implies $B \equiv \forall X \leq S : K. C$, $\Gamma, X \leq S : K \vdash_n A \leq C$, and $\Gamma \vdash \forall X \leq S : K. A : \star$.
4. $\Gamma \vdash_n \Lambda X : K. A \leq B$ implies $B \equiv \Lambda X : K. C$ and $\Gamma, X \leq \top^K : K \vdash_n A \leq C$.
5. $\Gamma \vdash_n AS \leq B$ implies $B \equiv AS$, or $\Gamma \vdash_n (\text{lub}_\Gamma(AS))^{nf} \leq B$, and $\Gamma \vdash AS : K$.
6. $\Gamma \vdash_n \bigwedge^K [A_1 \dots A_m] \leq B$ implies that there exists $j \in \{1..m\}$ such that $\Gamma \vdash_n A_j \leq B$ and $\forall k \in \{1..m\} \Gamma \vdash A_k : K$.
7. $\Gamma \vdash_n A \leq \bigwedge^K [B_1 \dots B_n]$ implies that for each $i \in \{1..n\}$ $\Gamma \vdash_n A \leq B_i$ and $\Gamma \vdash A : K$.
8. $\Gamma \vdash_n \bigwedge^K [A_1 \dots A_m] \leq \bigwedge^K [B_1 \dots B_n]$ implies that for each $i \in \{1..n\}$ there exists $j \in \{1..m\}$ such that $\Gamma \vdash_n A_j \leq B_i$ and $\forall k \in \{1..m\} \Gamma \vdash A_k : K$.

Moreover, given a normal proof of any of the antecedents, the proofs of the consequents are proper subderivations.

PROOF: In each case, given a proof of the antecedent, there is also a proof in normal form. Due to lemma 3.56(2), such a derivation cannot end with an application of NS-TRANS, and, because of lemma 3.56(1), if it ends with NS-REFL, then it is a derivation of a subtyping judgement between type variables or type applications. Finally, the result follows by inspection of the other rules. \square

LEMMA 3.59

1. $\Gamma \vdash_n T \leq \bigwedge^K [A_1 \dots A_n]$ if and only if $\Gamma \vdash_n T \leq A_k$ for each $k \in \{1..n\}$.
2. $\Gamma \vdash_n T \leq \bigwedge^K [A_1 \dots A_n]$ if and only if $\Gamma \vdash_n T \leq \bigwedge^K [A_k]$ for each $k \in \{1..n\}$.
3. Let $\Gamma \vdash \bigwedge^K [A_1 \dots A_n] : K$. Then $\Gamma \vdash_n \bigwedge^K [A_1 \dots A_n] \leq T$ if and only if $\Gamma \vdash_n A_k \leq T$ for some $k \in \{1..n\}$.

PROOF: By induction on derivations, using lemma 3.55 and generation. \square

3.6 Equivalence of ordinary and normal subtyping

In this section, we show that a subtyping judgement is derivable in F_{\wedge}^{ω} if and only if the corresponding normalized judgement is derivable in NF_{\wedge}^{ω} . This equivalence is proved in theorem 3.68. As usual, we need some auxiliary properties and definitions, among which we can highlight propositions 3.61 (Soundness) and 3.67 (Completeness).

LEMMA 3.60 Let $\text{lub}_{\Gamma}(S)$ be defined. Then

1. $\Gamma \vdash S : K$ implies $\Gamma \vdash \text{lub}_{\Gamma}(S) : K$.
2. $\Gamma \vdash S \leq \text{lub}_{\Gamma}(S)$.

PROOF: Item 1 follows by induction on derivations, while item 2 follows by induction on the structure of S . \square

PROPOSITION 3.61 (*Soundness*) If $\Gamma \vdash_n S \leq T$, then $\Gamma \vdash S \leq T$.

PROOF: By induction on the derivation of $\Gamma \vdash_n S \leq T$. We consider here a few illustrative cases.

NS-TVAR By the induction hypothesis, S-TVAR and S-TRANS.

NS-OAPP By the induction hypothesis, lemma 3.60(2), S-CONV and S-TRANS.

NS- $\forall\exists$ We are given that for each k in $\{1..n\}$ $\Gamma \vdash A_k : K$, and for each i in $\{1..m\}$ there is a j in $\{1..n\}$ such that $\Gamma \vdash_n A_j \leq B_i$. By K-MEET, $\Gamma \vdash \bigwedge^K [A_1..A_n] : K$, and, by S-MEET-LB, $\Gamma \vdash \bigwedge^K [A_1..A_n] \leq A_k$ for each k . Hence the result follows by the induction hypothesis, S-TRANS and S-MEET-G. \square

Note that the cases for type variable and type application reveal the fact that NS-TVAR and NS-OAPP hide steps of transitivity.

The following lemma says that empty intersections, \top^K , are maximal elements of the subtyping order.

LEMMA 3.62

1. $\Gamma \vdash T : K$ implies $\Gamma \vdash_n T \leq \top^K$.
2. $\Gamma \vdash T : K$ implies $\Gamma \vdash T \leq \top^K$.

PROOF: Statement 1 follows by the cases $m = 0$ in NS- \forall and NS- $\forall\exists$. Statement 2 is the case $n = 0$ in S-MEET-G. \square

LEMMA 3.63

1. $\Gamma \vdash \text{ok}$ implies $\Gamma^{nf} \vdash \text{ok}$.
2. $\Gamma \vdash T : K$ implies $\Gamma^{nf} \vdash T : K$.
3. $\Gamma \vdash S \leq T$ implies $\Gamma^{nf} \vdash S \leq T$.
4. Let $\Gamma_1, \Gamma_2 \vdash \text{ok}$. Then $\Gamma_1^{nf}, \Gamma_2 \vdash T : K$ implies $\Gamma_1, \Gamma_2 \vdash T : K$.
5. Let $\Gamma_1, \Gamma_2 \vdash \text{ok}$. Then $\Gamma_1^{nf}, \Gamma_2 \vdash S \leq T$ implies $\Gamma_1, \Gamma_2 \vdash S \leq T$.
6. Let $\Gamma \vdash S, T : K$. Then $\Gamma^{nf} \vdash S^{nf} \leq T^{nf}$ if and only if $\Gamma \vdash S \leq T$.

PROOF: Statements 1 and 2 follow by simultaneous induction on the size of derivations using lemma 3.31. Statement 3 follows by induction on the derivation of $\Gamma \vdash S \leq T$ using part 1, part 2, and lemma 3.31. Statement 4 follows by induction on the derivation of $\Gamma_1^{nf}, \Gamma_2 \vdash T : K$. Item 5 follows by induction on the derivation of $\Gamma_1^{nf}, \Gamma_2 \vdash S \leq T$, using part 4. Item 6 is a corollary of part 3, part 5 and lemma 3.30. \square

In the last lemma, items 1, 2, and 3 show that well formation of contexts, kinding judgements, and subtyping judgements are invariant under normalization of contexts, while items 4 and 5 are the converse of 2 and 3 respectively.

The following lemma states that S-TVAR is an admissible rule in NF_{λ}^{ω} .

LEMMA 3.64 Let Γ be a context in normal form such that $\Gamma \vdash \text{ok}$ and $Y \in \text{dom}(\Gamma)$. Then $\Gamma \vdash_n Y \leq \Gamma(Y)$.

PROOF: Let $\Gamma \equiv \Gamma_1, Y \leq T : K, \Gamma_2$. By lemma 3.16, $\Gamma_1 \vdash T : K$. If T is not an intersection, then, by NS-REFL and NS-TVAR, we have $\Gamma \vdash_n Y \leq T$. If $T \equiv \bigwedge^{K'} [B_1..B_m]$, then by generation for kinding and uniqueness of kinds, $\Gamma \vdash B_i : K$ for each i and $K \equiv K'$. By NS-REFL, $\Gamma \vdash_n B_i \leq B_i$ for each i . Then, by NS- \exists and NS-TVAR, it follows that $\Gamma \vdash_n Y \leq B_i$ for each i , and, by NS- \forall , $\Gamma \vdash_n Y \leq T$. \square

The following lemma shows that the normal subtyping system has the substitution property.

LEMMA 3.65 (*Substitution*) If $\Gamma \vdash U : K$ and $\Gamma, X : K, \Gamma' \vdash_n S \leq T$, then $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n (S[X \leftarrow U])^{nf} \leq (T[X \leftarrow U])^{nf}$.

PROOF: By induction on the derivation of $\Gamma, X : K, \Gamma' \vdash_n S \leq T$. For the sake of clarity, we sometimes leave out kinding judgements and their justifications which follow easily from the structural properties in section 3.2. We show here a couple of representative cases. Let $\Gamma'' \equiv \Gamma, X : K, \Gamma'$.

NS-TVAR We are given $\Gamma'' \vdash_n \Gamma''(Y) \leq A$. We have to consider three cases.

1. $Y \equiv X$. By subject reduction, $\Gamma \vdash U^{nf} : K$, and by lemma 3.62(1), it follows that $\Gamma \vdash_n U^{nf} \leq \top^K$. By weakening, it follows that $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n U^{nf} \leq \top^K$ and, by the induction hypothesis, it follows that $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n \top^K \leq (A[X \leftarrow U])^{nf}$. Finally, the result follows by NS-TRANS.
2. $Y \in \text{dom}(\Gamma)$. By the free variables lemma, $X \notin \text{FV}(\Gamma(Y))$ and $X \neq Y$. By lemmas 3.25, 3.63(1), and 3.64, it follows that $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n Y \leq \Gamma(Y)$, and, by the induction hypothesis, it follows that $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n \Gamma(Y) \leq (A[X \leftarrow U])^{nf}$. Finally, the result follows by NS-TRANS.
3. $Y \in \text{dom}(\Gamma')$. By the induction hypothesis, it follows that

$$\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n (\Gamma'(Y)[X \leftarrow U])^{nf} \leq (A[X \leftarrow U])^{nf}.$$

By lemmas 3.25, 3.63(1), and 3.64, $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n Y \leq (\Gamma'[X \leftarrow U])^{nf}(Y)$. Furthermore, $(\Gamma, (\Gamma'[X \leftarrow U])^{nf})(Y) = (\Gamma'(Y)[X \leftarrow U])^{nf}$. Hence the result follows by NS-TRANS.

NS-ARROW We are given that $\Gamma'' \vdash_n T \leq S$ and $\Gamma'' \vdash_n A \leq B$. By the induction hypothesis, $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n (T[X \leftarrow U])^{nf} \leq (S[X \leftarrow U])^{nf}$ and $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n (A[X \leftarrow U])^{nf} \leq (B[X \leftarrow U])^{nf}$. There are four cases to consider, since $(A[X \leftarrow U])^{nf}$ and $(B[X \leftarrow U])^{nf}$ may be intersections or not. We shall consider only two of them to illustrate the proof method.

1. $(A[X \leftarrow U])^{nf}$ and $(B[X \leftarrow U])^{nf}$ are not intersections. Then the result follows by applying NS-ARROW.
2. $(A[X \leftarrow U])^{nf} = \bigwedge^* [C_1..C_n]$ and $(B[X \leftarrow U])^{nf}$ is not an intersection. Then we have that

$$\begin{aligned} ((T \rightarrow B)[X \leftarrow U])^{nf} &= (T[X \leftarrow U])^{nf} \rightarrow (B[X \leftarrow U])^{nf} \quad \text{and} \\ ((S \rightarrow A)[X \leftarrow U])^{nf} &= \bigwedge^* [(S[X \leftarrow U])^{nf} \rightarrow C_1..(S[X \leftarrow U])^{nf} \rightarrow C_n]. \end{aligned}$$

By lemma 3.58, it follows that for some i $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n C_i \leq (B[X \leftarrow U])^{nf}$. Applying NS-ARROW, $\Gamma, (\Gamma'[X \leftarrow U])^{nf} \vdash_n (S[X \leftarrow U])^{nf} \rightarrow C_i \leq (T[X \leftarrow U])^{nf} \rightarrow (B[X \leftarrow U])^{nf}$. Finally, the result follows by NS- \exists . \square

This substitution lemma is the key result we use in proving that S-OAPP has a corresponding admissible rule in NF_λ^ω .

LEMMA 3.66 $\Gamma \vdash SU : K$. Then $\Gamma \vdash_n S \leq T$ implies $\Gamma \vdash_n (SU)^{nf} \leq (TU)^{nf}$.

PROOF: By induction on the derivation of $\Gamma \vdash_n S \leq T$, assuming a derivation in normal form. The cases for NS-ARROW and NS-ALL cannot occur, because of the assumption $\Gamma \vdash SU : K$. We show here the interesting cases.

NS-TVAR We are given $\Gamma \vdash_n \Gamma(X) \leq A$. By the induction hypothesis, $\Gamma \vdash_n (\Gamma(X)U)^{nf} \leq (AU)^{nf}$. We have to consider two cases.

$(AU)^{nf} \equiv B$ By NS-OAPP.

$(AU)^{nf} \equiv \bigwedge^K [A_1 \dots A_n]$ By lemma 3.59, $\Gamma \vdash_n (\Gamma(X)U)^{nf} \leq A_k$ for each k in $\{1..n\}$. By NS-OAPP, $\Gamma \vdash_n XU \leq (A_k)$ for each k , which, by NS- \forall , implies $\Gamma \vdash_n XU \leq (AU)^{nf}$.

NS-OABS We are given $\Gamma, X:K \vdash_n A \leq B$. By the substitution lemma 3.65, it follows that $\Gamma \vdash_n (A[X \leftarrow U])^{nf} \leq (B[X \leftarrow U])^{nf}$. On the other hand, we have that $(\lambda X:K.A)U \rightarrow_{\beta\wedge} A[X \leftarrow U]$ and $(\lambda X:K.B)U \rightarrow_{\beta\wedge} B[X \leftarrow U]$. Finally, the result follows by the uniqueness of normal forms. \square

PROPOSITION 3.67 (*Completeness*) If $\Gamma \vdash S \leq T$, then $\Gamma^{nf} \vdash_n S^{nf} \leq T^{nf}$.

PROOF: By induction on the derivation of $\Gamma \vdash S \leq T$, using lemma 3.66 for the case of S-OAPP, and lemma 3.63(2), strong normalization for $\rightarrow_{\beta\wedge}$ (theorem 3.43, Church-Rosser for $\rightarrow_{\beta\wedge}$ (theorem 3.5), uniqueness of normal forms (corollary 3.12), subject reduction for kinding (3.26), and NS-REFL for the case S-CONV \square

THEOREM 3.68 (*Equivalence of ordinary and normal subtyping*) Let $\Gamma \vdash S : K$ and $\Gamma \vdash T : K$. Then $\Gamma \vdash S \leq T$ if and only if $\Gamma^{nf} \vdash_n S^{nf} \leq T^{nf}$.

PROOF: \Rightarrow) By completeness (3.67). \Leftarrow) By soundness (3.61), it follows that $\Gamma^{nf} \vdash_n S^{nf} \leq T^{nf}$, and, by lemma 3.63(6), it follows that $\Gamma \vdash S \leq T$. \square

3.6.1 Least strict upper bound

So far we only used that $\text{lub}_\Gamma(S)$ is an upper bound of S in the context Γ (See lemma 3.60(2)). We can now give the final motivation of the name we chose, showing that if $\text{lub}_\Gamma(S)$ is defined and $T \neq_{\beta\wedge} S$, then $\Gamma \vdash S \leq T$ implies $\Gamma \vdash \text{lub}_\Gamma(S) \leq T$. We first show that the corresponding property holds for the normalized system.

LEMMA 3.69 Let $\text{lub}_\Gamma(S)$ be defined. Then

1. If $S \twoheadrightarrow_{\beta\wedge} S'$ and $\Gamma \twoheadrightarrow_{\beta\wedge} \Gamma'$, then $\text{lub}_\Gamma(S) \twoheadrightarrow_{\beta\wedge} \text{lub}_{\Gamma'}(S')$.
2. If $\Gamma \vdash_n S \leq T$, then $\Gamma \vdash_n \text{lub}_\Gamma(S)^{nf} \leq T$ or $S \equiv T$.

PROOF:

1. By induction on the structure of S , observing that if $\text{lub}_\Gamma(S)$ is defined, so is $\text{lub}_{\Gamma'}(S')$.
2. By induction on the derivation of $\Gamma \vdash_n S \leq T$. It is immediate for the case NS-REFL; for NS-ARROW, NS-ALL, and NS-OABS $\text{lub}_\Gamma(S)$ is not defined; for the other rules the result follows using the induction hypothesis. \square

COROLLARY 3.70 Let $\text{lub}_\Gamma(S)$ be defined. Then $\Gamma \vdash S \leq T$ and $T \not\equiv_{\beta\wedge} S$ implies $\Gamma \vdash \text{lub}_\Gamma(S) \leq T$.

PROOF: By completeness, it follows that $\Gamma^{nf} \vdash_n S^{nf} \leq T^{nf}$. By lemma 3.69(2), $\Gamma^{nf} \vdash_n (\text{lub}_{\Gamma^{nf}}(S^{nf}))^{nf} \leq T^{nf}$, because $S^{nf} \not\equiv T^{nf}$. By soundness, it follows that $\Gamma^{nf} \vdash (\text{lub}_{\Gamma^{nf}}(S^{nf}))^{nf} \leq T^{nf}$, which is equivalent to $\Gamma \vdash \text{lub}_{\Gamma^{nf}}(S^{nf}) \leq T$ by lemma 3.63(6). Finally, (using lemmas 3.60(1) and 3.26, and proposition 3.32 to get the corresponding kinding judgements) it follows that $\Gamma \vdash \text{lub}_\Gamma(S) \leq T$ by lemma 3.69(1), S-CONV and S-TRANS. \square

3.7 A subtype checking algorithm, $\text{Alg}F_\wedge^\omega$

As it stands, NF_\wedge^ω as defined in section 3.4.1 is not a deterministic algorithm, because its rules are not syntax directed. Fortunately, we are not far away from an algorithmic presentation. In fact, corollary 3.57 is the bridge to the algorithmic presentation of the subtyping relation, $\text{Alg}F_\wedge^\omega$, which states that transitivity steps can be eliminated and reflexivity steps can be simplified. $\text{Alg}F_\wedge^\omega$ is obtained from NF_\wedge^ω by removing NS-TRANS and restricting NS-REFL to type variables and type applications.

DEFINITION 3.71 ($\text{Alg}F_\wedge^\omega$ subtyping rules)

$$\begin{array}{c} \frac{\Gamma \vdash X : K}{\Gamma \vdash_{\text{Alg}} X \leq X} \quad (\text{ALGS-TVARREFL}) \\ \\ \frac{\Gamma \vdash TS : K}{\Gamma \vdash_{\text{Alg}} TS \leq TS} \quad (\text{ALGS-OAPPREFL}) \\ \\ \frac{\Gamma \vdash_{\text{Alg}} \Gamma(X) \leq A \quad X \not\equiv A}{\Gamma \vdash_{\text{Alg}} X \leq A} \quad (\text{ALGS-TVAR}) \\ \\ \frac{\Gamma \vdash_{\text{Alg}} T \leq S \quad \Gamma \vdash_{\text{Alg}} A \leq B \quad \Gamma \vdash S \rightarrow A : \star}{\Gamma \vdash_{\text{Alg}} S \rightarrow A \leq T \rightarrow B} \quad (\text{ALGS-ARROW}) \\ \\ \frac{\Gamma, X \leq S : K \vdash_{\text{Alg}} A \leq B \quad \Gamma \vdash \forall X \leq S : K. A : \star}{\Gamma \vdash_{\text{Alg}} \forall X \leq S : K. A \leq \forall X \leq S : K. B} \quad (\text{ALGS-ALL}) \\ \\ \frac{\Gamma, X \leq \top^K : K \vdash_{\text{Alg}} A \leq B}{\Gamma \vdash_{\text{Alg}} \Lambda X : K. A \leq \Lambda X : K. B} \quad (\text{ALGS-OABS}) \\ \\ \frac{\Gamma \vdash_{\text{Alg}} (\text{lub}_\Gamma(TS))^{nf} \leq A \quad \Gamma \vdash TS : K \quad TS \not\equiv A}{\Gamma \vdash_{\text{Alg}} TS \leq A} \quad (\text{ALGS-OAPP}) \\ \\ \frac{\forall i \in \{1..m\} \Gamma \vdash_{\text{Alg}} A \leq A_i \quad \Gamma \vdash A : K}{\Gamma \vdash_{\text{Alg}} A \leq \bigwedge^K [A_1..A_m]} \quad (\text{ALGS-V}) \\ \\ \frac{\exists j \in \{1..n\} \Gamma \vdash_{\text{Alg}} A_j \leq A \quad \forall k \in \{1..n\} \Gamma \vdash A_k : K}{\Gamma \vdash_{\text{Alg}} \bigwedge^K [A_1..A_n] \leq A} \quad (\text{ALGS-E}) \\ \\ \frac{\forall i \in \{1..m\} \exists j \in \{1..n\} \Gamma \vdash_{\text{Alg}} A_j \leq B_i \quad \forall k \in \{1..n\} \Gamma \vdash A_k : K}{\Gamma \vdash_{\text{Alg}} \bigwedge^K [A_1..A_n] \leq \bigwedge^K [B_1..B_m]} \quad (\text{ALGS-V E}) \end{array}$$

We assume that if $\Gamma \vdash_{\text{Alg}} S \leq T$, then S, T , and all types appearing in Γ are in $\beta\wedge$ -normal form. Furthermore, A, B , and C range over types whose outermost constructor is not an intersection.

REMARK 3.72 It is an immediate consequence of the $\beta\wedge$ -reduction rules that, if T is in $\beta\wedge$ -normal form, then T is either $X, S \rightarrow A, \forall X \leq S : K. A, \Lambda X : K. A, A S$ where A is not an abstraction, or $\bigwedge^K [A_1..A_n]$. We frequently use this notation as a reminder of the shape of types in normal form.

LEMMA 3.73 (*Equivalence of normal and algorithmic subtyping*)

Let $\Gamma \vdash S, T : K$. Then $\Gamma \vdash_n S \leq T$ if and only if $\Gamma \vdash_{Alg} S \leq T$.

PROOF: (\Rightarrow) By corollary 3.57. (\Leftarrow) Immediate. \square

We have thereby proved that $AlgF_\lambda^\omega$ is indeed a sound and complete algorithm to compute F_λ^ω 's subtyping relation.

PROPOSITION 3.74 (*Equivalence of ordinary and algorithmic subtyping*)

Let $\Gamma \vdash S : K$ and $\Gamma \vdash T : K$. Then $\Gamma \vdash S \leq T$ if and only if $\Gamma^{nf} \vdash_{Alg} S^{nf} \leq T^{nf}$.

PROOF: By the equivalence of ordinary and normal subtyping (theorem 3.68) and the equivalence of normal and algorithmic subtyping (lemma 3.73). \square

3.7.1 Example

In this section, we give the derivation in $AlgF_\lambda^\omega$ of the example 3.44 (also mentioned in the introduction). Let $\Gamma \equiv W : K, X \leq \Lambda Y : K.Y : K \rightarrow K, Z \leq X : K \rightarrow K$. We present a proof in the normal system of $\Gamma^{nf} \vdash_{Alg} X(ZW)^{nf} \leq W^{nf}$, which is the translation of $\Gamma \vdash X(ZW) \leq W$.

Observe that $\Gamma^{nf} \equiv \Gamma$,

$$(X(ZW))^{nf} \equiv X(ZW), \quad \text{and} \\ W^{nf} \equiv W.$$

For the sake of readability we omit kinding judgements. The derivation in normal form in $AlgF_\lambda^\omega$ is substantially shorter than the one in F_λ^ω shown in section 4.2.

$$\frac{\frac{\frac{\Gamma \vdash W : K}{\Gamma \vdash_{Alg} ((\Lambda Y : K.Y)W)^{nf} \leq W} \text{AS-REFL}}{\Gamma \vdash_{Alg} XW \leq W} \text{AS-OAPP}}{\Gamma \vdash_{Alg} ((\Lambda Y : K.Y)(ZW))^{nf} \leq W} \text{AS-OAPP}}{\Gamma \vdash_{Alg} X(ZW) \leq W} \text{AS-OAPP}$$

3.8 Type checking and type reconstruction

Given a context Γ , a term e , and a type T , type checking consists of analyzing whether the judgement $\Gamma \vdash e : T$ is derivable from a given set of inference rules. Type checking algorithms for lambda calculi, unless they are formulated using Gentzen's sequent calculus style, involve *guessing* the type of subterms. For example, when e is $e_1 e_2$, the type of e_2 is not necessarily a subexpression of T , and in order to corroborate or to refute the assertion $\Gamma \vdash e : T$ we need to *infer* a type for e_2 .

In this section, we present an algorithm for inferring minimal types in F_λ^ω . Given Γ and e , the type S constructed by the algorithm is a subtype of every T such that $\Gamma \vdash e : T$. In this way, we reduce the problem of whether $\Gamma \vdash e : T$ to that of inferring a type S such that $\Gamma \vdash e : S$ and $\Gamma \vdash S \leq T$. Solving this problem involves not only the typing rules but all the inference rules of F_λ^ω : the rule T-SUBSUMPTION depends on a subtyping judgement, the rule T-VAR depends on an ok judgement, and the ok judgements depend on kinding judgements. Consequently, type checking uses the full power of the F_λ^ω system.

As an example, consider type checking the following judgement:

$$\Gamma, X \leq T_1 \rightarrow T_2, f : X, a : T_1 \vdash f a : T_2.$$

The application $f a$ can only be formed if f has an arrow type. Using T-VAR we can assign type X to f , which means that in order to obtain an arrow type for f we have to *replace* X by its bound, which has the right form. Observe how this *replacement* is performed by T-SUBSUMPTION in the following derivation.

$$\begin{array}{c}
\frac{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash \text{ok} \quad \Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash \text{ok}}{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash \boxed{f : X} \quad \Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash X \leq T_1 \rightarrow T_2} \text{T-SUB} \\
\frac{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash \boxed{f : T_1 \rightarrow T_2}}{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash \text{ok}} \\
\frac{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash f : T_1 \rightarrow T_2 \quad \Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash a : T_1}{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:T_1 \vdash f a : T_2} \text{T-APP}
\end{array}$$

Note that, in the presence of T-SUBSUMPTION, we may actually perform the application when the type of a is a subtype of T_1 . Namely, if

$$\frac{\Gamma, X \leq T_1 \rightarrow T_2, f:X, \boxed{a:U_1} \vdash a : U_1 \quad \Gamma, X \leq T_1 \rightarrow T_2, f:X, a:U_1 \vdash U_1 \leq T_1}{\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:U_1 \vdash \boxed{a : T_1}} \text{T-SUB}$$

Moreover, we may want to check whether $\Gamma, X \leq T_1 \rightarrow T_2, f:X, a:U_1 \vdash f a : U_2$, where T_2 is a subtype of U_2 .

The situation gets more complicated if f has an intersection type. Suppose that

$$\Gamma, X \leq T_1 \rightarrow T_2, Y \leq S_1 \rightarrow S_2, f:X \wedge Y \wedge \forall Z \leq V_1 : K.V_2, a:U_1 \vdash f a : U_2,$$

where U_1 is a subtype of T_1 and S_1 . An algorithm should not consider the type $\forall Z \leq V_1 : K.V_2$ for f since, in this case, f is applied to a term and not to a type. Then it has to replace X and Y by their bounds, $T_1 \rightarrow T_2$ and $S_1 \rightarrow S_2$. Moreover, given that the type of a , U_1 , is a subtype of both S_1 and T_1 , it should check whether $S_2 \wedge T_2$ is a subtype of U_2 .

Another source of problems in the search for an algorithmic presentation of the typing rules is that types may not be in normal form. Consider the judgement

$$\Gamma, X \leq T_1 \rightarrow T_2, Z \leq \Lambda Y : \star.Y, f:Z X, a:T_1 \vdash f a : T_2, \quad (1)$$

In order to type the application, f should be assigned type $T_1 \rightarrow T_2$. To do that, Z should be replaced by its bound in $Z X$. This replacement produces a type which is not in normal form, so $\Lambda Y : \star.Y X$ has to be normalized to obtain X . Finally, X is replaced by its bound and then the application can be typed.

The main new source of difficulty is the interaction between the need for normalization and the presence of intersection types.

An algorithm to infer types should proceed structurally on the form of the term whose type is to be inferred. This requires us to remove the rules which make our typing rules non-deterministic: we should eliminate T-SUBSUMPTION and T-MEET from the original presentation, and modify the other rules in such a way that we can still type the same set of terms.

We give some preliminary definitions and results before presenting the rules of our new system:

- We define the mapping *flub*, which performs the “replacements” which we motivated with the previous examples.
- We define the function *arrows*, to filter arrow types in order to deal with term application.
- We define the function *alls* to filter polymorphic types to deal with type application.

The function *lub* (definition 3.47) is a partial function which is only defined for type variables and type applications. Here, we extend the definition of *lub* to intersection types in such a way that it is defined if the least upper bound is defined for at least one of the types in the intersection.

DEFINITION 3.75 (*Homomorphic extension of lub to intersections, lub**)

$$\begin{aligned}
\text{lub}_\Gamma^*(X) &= \Gamma(X), \\
\text{lub}_\Gamma^*(ST) &= \text{lub}_\Gamma^*(S)T, \\
\text{lub}_\Gamma^*(\bigwedge^K [T_1..T_n]) &= \bigwedge^K [T'_1..T'_n], \quad \text{if } \exists i \in \{1..n\} \text{ such that } \text{lub}_\Gamma^*(T_i) \downarrow,
\end{aligned}$$

where T'_i is $\text{lub}_\Gamma^*(T_i)$, if $\text{lub}_\Gamma^*(T_i) \downarrow$, and T_i otherwise, and $T \downarrow$ means T is defined.

LEMMA 3.76 If $\text{lub}_\Gamma^*(T)$ is defined, then $\Gamma \vdash T \leq \text{lub}_\Gamma^*(T)$.

PROOF: By induction on the complexity of T , using corollary 3.29. \square

We define the mapping flub which given a type T (and a context Γ) finds the smallest type larger than T (with respect to the subtype relation) having structural information to perform an application.

DEFINITION 3.77 (*Functional Least Upper Bound*) The functional least upper bound of a type T , in a context Γ , $\text{flub}_\Gamma(T)$ is defined as follows.

$$\text{flub}_\Gamma(T) = \begin{cases} \text{flub}_\Gamma(\text{lub}_\Gamma^*(T^{nf})), & \text{if } \text{lub}_\Gamma^*(T^{nf}) \downarrow; \\ T^{nf}, & \text{otherwise.}^1 \end{cases}$$

The intuition behind the definition of the function flub is to find $T_1 \rightarrow T_2$ starting from $Z X$ in the example 1 above. In other words, $\text{flub}_\Gamma(Z X) = T_1 \rightarrow T_2$. For simplicity we assume $T_1 \rightarrow T_2$ in normal form. Step by step,

$$\begin{aligned} \text{flub}_\Gamma(Z X) &= \text{flub}_\Gamma(\text{lub}_\Gamma^*(Z X)) \\ &= \text{flub}_\Gamma((\Lambda Y:K.Y) X) \\ &= \text{flub}_\Gamma(\text{lub}_\Gamma^*((\Lambda Y:K.Y) X)^{nf}) \\ &= \text{flub}_\Gamma(\text{lub}_\Gamma^*(X)) \\ &= \text{flub}_\Gamma(T_1 \rightarrow T_2) \\ &= T_1 \rightarrow T_2. \end{aligned}$$

More generally, flub climbs the subtyping hierarchy until it finds an arrow, a quantifier, or an intersection of these two.

LEMMA 3.78 Let $\Gamma \vdash S, T : \star$ and $S =_{\beta\wedge} T$. Then $\text{flub}_\Gamma(S) \equiv \text{flub}_\Gamma(T)$.

DEFINITION 3.79 (*arrows and alls*)

1. $\text{arrows}(T_1 \rightarrow T_2) = \{T_1 \rightarrow T_2\},$
 $\text{arrows}(\bigwedge^*[T_1..T_n]) = \bigcup_{i \in \{1..n\}} \text{arrows}(T_i),$
 $\text{arrows}(T) = \emptyset, \quad \text{if } T \not\equiv T_1 \rightarrow T_2 \text{ and } T \not\equiv \bigwedge^*[T_1..T_n].$
2. $\text{alls}(\forall X \leq T_1:K.T_2) = \{\forall X \leq T_1:K.T_2\},$
 $\text{alls}(\bigwedge^*[T_1..T_n]) = \bigcup_{i \in \{1..n\}} \text{alls}(T_i),$
 $\text{alls}(T) = \emptyset, \quad \text{if } T \not\equiv \forall X \leq T_1:K.T_2 \text{ and } T \not\equiv \bigwedge^*[T_1..T_n].$

The situation here is significantly more complex than in [47] for F_\wedge , an extension of the second order λ -calculus. There it is enough to recursively search for arrows or polymorphic types in the context, because in F_\wedge there is no reduction on types. The information to be searched for is explicit in the context, so the job done here by flub is simply an extra case in the definition of arrows and alls . Namely,

$$\begin{aligned} \text{arrows}(X) &= \text{arrows}(\Gamma(X)) \quad \text{and} \\ \text{alls}(X) &= \text{alls}(\Gamma(X)). \end{aligned}$$

Moreover, to prove that flub is well-founded is similar for us in complexity to proving termination of subtype checking. The similarity comes from the fact that computing flub involves replacing variables by their bounds in a given context and normalizing with respect to $\rightarrow_{\beta\wedge}$. The well-foundedness of flub is shown in lemma 4.24. In contrast, in [47] it is enough to observe that well-formed contexts cannot contain cycles of variable references.

¹This step can be optimised in an implementation of the type checking algorithm, allowing us to avoid the normalization of T when T is either an arrow type or a quantified type.

NOTATION 3.80 We introduce a new notation for intersection types. The intersection of all types T such that $\phi(T)$ holds is written $\bigwedge^K [T \mid \phi(T)]$. Note that this is an alternative notation to $\bigwedge^K [T_1..T_n]$ such that $\phi(T_i)$ holds if and only if $i \in \{1..n\}$.

We can now define a type inference algorithm for F_\wedge^ω .

DEFINITION 3.81 (A type inference algorithm, *inf*)

$$\frac{\Gamma_1, x:T, \Gamma_2 \vdash \text{ok}}{\Gamma_1, x:T, \Gamma_2 \vdash_{\text{inf}} x : T} \quad (\text{AT-VAR})$$

$$\frac{\Gamma, x:T_1 \vdash_{\text{inf}} e : T_2}{\Gamma \vdash_{\text{inf}} \lambda x:T_1. e : T_1 \rightarrow T_2} \quad (\text{AT-ABS})$$

$$\frac{\Gamma \vdash_{\text{inf}} f : T \quad \Gamma \vdash_{\text{inf}} a : S}{\Gamma \vdash_{\text{inf}} f a : \bigwedge^* [T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i]} \quad (\text{AT-APP})$$

$$\frac{\Gamma, X \leq T_1 : K_1 \vdash_{\text{inf}} e : T_2}{\Gamma \vdash_{\text{inf}} \lambda X \leq T_1 : K_1. e : \forall X \leq T_1 : K_1. T_2} \quad (\text{AT-TABS})$$

$$\frac{\Gamma \vdash_{\text{inf}} f : T}{\Gamma \vdash_{\text{inf}} f S : \bigwedge^* [T_i [X \leftarrow S] \mid \forall X \leq S_i : K. T_i \in \text{alls}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i]} \quad (\text{AT-TAPP})$$

$$\frac{\text{for all } i \in \{1..n\} \quad \Gamma \vdash_{\text{inf}} e [X \leftarrow S_i] \in T_i}{\Gamma \vdash_{\text{inf}} \text{for}(X \in S_1..S_n) e : \bigwedge^* [T_1..T_n]} \quad (\text{AT-FOR})$$

The algorithmic information of rule AT-APP is that in order to find a type for $f a$ in Γ , we need to infer a type S for a and a type T for f , and to take the intersection of all the T_i 's such that $T_i \rightarrow S_i \in \text{arrows}(\text{flub}_\Gamma(T))$ and $\Gamma \vdash S \leq S_i$. Note that since $\text{arrows}(U)$ is a finite set for any type U , the intersection is finite. Similarly, $\text{alls}(U)$ is a finite set for any type U therefore the intersection in the conclusion of AT-TAPP is finite.

3.9 Minimal typing

In this section we show that F_\wedge^ω satisfies the minimal typing property (theorem 3.92). We first prove that the algorithm *inf* is sound with respect to F_\wedge^ω : if $\Gamma \vdash_{\text{inf}} e : T$, then $\Gamma \vdash e : T$ (proposition 3.85). We then prove that every closed term is typeable using either set of typing rules (lemma 3.89). Finally, we prove that *inf* computes minimal types for F_\wedge^ω terms (proposition 3.91).

LEMMA 3.82 Let $\Gamma \vdash T : \star$. Then $\Gamma \vdash T \leq \text{flub}_\Gamma(T)$.

PROOF: Since *flub* is well-founded, we can proceed by induction on the number of unfolding steps in $\text{flub}_\Gamma(T)$. If $\text{flub}_\Gamma(T) = T^{nf}$, the result follows by S-CONV. Otherwise, $\text{flub}_\Gamma(T) = \text{flub}_\Gamma(\text{lub}_\Gamma^*(T^{nf}))$. By S-CONV, $\Gamma \vdash T \leq T^{nf}$. By lemma 4.22, $\Gamma \vdash T^{nf} \leq \text{lub}_\Gamma^*(T^{nf})$. By the induction hypothesis, $\Gamma \vdash \text{lub}_\Gamma^*(T^{nf}) \leq \text{flub}_\Gamma(\text{lub}_\Gamma^*(T^{nf}))$. Finally, by S-TRANS, the result follows. \square

LEMMA 3.83 Let $\Gamma \vdash T : \star$. Then

1. $\Gamma \vdash T \leq \bigwedge^* [S \mid S \in \text{arrows}(\text{flub}_\Gamma(T))]$.
2. $\Gamma \vdash T \leq \bigwedge^* [S \mid S \in \text{alls}(\text{flub}_\Gamma(T))]$.

PROOF: Item 1: Using lemma 3.82, we reduce our problem to proving that

$$\Gamma \vdash T \leq \bigwedge^* [S \mid S \in \text{arrows}(T)],$$

which follows by induction on the structure of T . Item 2 follows similarly. \square

LEMMA 3.84

1. If $\Gamma \vdash T \leq T_1 \rightarrow T_2$, then $\Gamma \vdash \bigwedge^*[S \mid S \in \text{arrows}(\text{flub}_\Gamma(T))] \leq T_1 \rightarrow T_2$.
2. If $\Gamma \vdash T \leq \forall X \leq T_1 : K.T_2$, then $\Gamma \vdash \bigwedge^*[S \mid S \in \text{alls}(\text{flub}_\Gamma(T))] \leq \forall X \leq T_1 : K.T_2$.

PROOF: We show only case 1 here, case 2 is similar. By induction on the derivation of $\Gamma \vdash T \leq T_1 \rightarrow T_2$. The last rule of a derivation of this subtyping judgement can only be S-CONV, S-TVAR, S-TRANS, or S-MEET-LB. The first three cases use similar arguments, therefore we consider here only the cases for S-CONV and S-MEET-LB.

S-CONV We are given that $T =_{\beta\wedge} T_1 \rightarrow T_2$. By lemma 4.25 and the definition of *flub*, we have that:

$$\text{arrows}(\text{flub}_\Gamma(T)) = \text{arrows}(\text{flub}_\Gamma(T_1 \rightarrow T_2)) = \text{arrows}((T_1 \rightarrow T_2)^{nf})$$

We now have two cases to analyze.

1. If $(T_1 \rightarrow T_2)^{nf} = T_1^{nf} \rightarrow T_2^{nf}$, then the result follows by S-MEET-LB and S-CONV.
2. Otherwise, let $(T_1 \rightarrow T_2)^{nf} = \bigwedge^*[T_1^{nf} \rightarrow U_1 .. T_1^{nf} \rightarrow U_n]$, where $T_2^{nf} = \bigwedge^*[U_1 .. U_n]$. Then, $\text{arrows}(\text{flub}_\Gamma(T)) = \{T_1^{nf} \rightarrow U_1 .. T_1^{nf} \rightarrow U_n\}$. Consequently, $\bigwedge^*[S \mid S \in \text{arrows}(\text{flub}_\Gamma(T))] = (T_1 \rightarrow T_2)^{nf}$, and the result follows by S-CONV.

S-MEET-LB We are given that $\Gamma \vdash \bigwedge^*[S_1 .. T_1 \rightarrow T_2 .. S_n] \leq T_1 \rightarrow T_2$. By the definition of *flub*, $\text{flub}_\Gamma(\bigwedge^*[S_1 .. T_1 \rightarrow T_2 .. S_n]) = \bigwedge^*[\dots T_1^{nf} \rightarrow A_1 .. T_1^{nf} \rightarrow A_m \dots]$, where $T_2^{nf} = \bigwedge^*[A_1 .. A_m]$ or $T_2^{nf} = A_1$. Now, $\text{arrows}(\text{flub}_\Gamma(\bigwedge^*[S_1 .. T_1 \rightarrow T_2 .. S_n])) \supseteq \{T_1^{nf} \rightarrow A_1 .. T_1^{nf} \rightarrow A_m\}$. Then, if $T_2^{nf} = \bigwedge^*[A_1 .. A_m]$, by lemma 3.28; and, if $T_2^{nf} = A_1$, by S-MEET-LB, we have that

$$\Gamma \vdash \bigwedge^*[S \mid S \in \text{arrows}(\text{flub}_\Gamma(\bigwedge^*[S_1 .. T_1 \rightarrow T_2 .. S_n]))] \leq (T_1 \rightarrow T_2)^{nf}.$$

Finally, the result follows by S-CONV. \square

PROPOSITION 3.85 (*Soundness of inf*) If $\Gamma \vdash_{\text{inf}} e : T$, then $\Gamma \vdash e : T$.

PROOF: By induction on the derivation of $\Gamma \vdash_{\text{inf}} e : T$. The interesting cases are when the last applied rule is either AT-APP and AT-TAPP.

AT-APP $\Gamma \vdash_{\text{inf}} f a : \bigwedge^*[T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i]$ is derived from $\Gamma \vdash_{\text{inf}} f : T$ and $\Gamma \vdash_{\text{inf}} a : S$. If $\bigwedge^*[T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i] =_{\beta\wedge} \top^*$, then the result follows immediately using T-MEET. Otherwise, by the induction hypothesis, we have that $\Gamma \vdash f : T$. By lemma 3.83(1), S-MEET-LB, S-TRANS, and T-SUBSUMPTION, $\Gamma \vdash f : S_i \rightarrow T_i$. By the induction hypothesis and T-SUBSUMPTION, $\Gamma \vdash a : S_i$. By T-APP, $\Gamma \vdash f a : T_i$. Finally, by T-MEET,

$$\Gamma \vdash f a : \bigwedge^*[T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i].$$

AT-TAPP $\Gamma \vdash_{\text{inf}} f S : \bigwedge^*[T_i[X \leftarrow S] \mid \forall X \leq S_i : K.T_i \in \text{alls}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i]$ is derived from $\Gamma \vdash_{\text{inf}} f : T$. If $\bigwedge^*[T_i[X \leftarrow S] \mid \forall X \leq S_i : K.T_i \in \text{alls}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i] =_{\beta\wedge} \top^*$, then the result follows immediately, using T-MEET. Otherwise, assume

$$\text{alls}(\text{flub}_\Gamma(T)) \equiv \{\forall X \leq S_1 : K.T_1 .. \forall X \leq S_n : K.T_n\}.$$

By the induction hypothesis, we have that, $\Gamma \vdash f : T$. By lemma 3.83(2), S-MEET-LB, S-TRANS, and T-SUBSUMPTION, it follows that

$$\Gamma \vdash f : \forall X \leq S_i : K.T_i. \text{ Since } \Gamma \vdash S \leq S_i, \text{ by T-APP, } \Gamma \vdash f S : T_i[X \leftarrow S]. \text{ Finally, by T-MEET, } \Gamma \vdash f S : \bigwedge^*[T_i[X \leftarrow S] \mid \forall X \leq S_i : K.T_i \in \text{alls}(\text{flub}_\Gamma(T)) \text{ and } \Gamma \vdash S \leq S_i]. \quad \square$$

LEMMA 3.86 (*Term application*)

If $\Gamma \vdash \bigwedge^*[S_1 \rightarrow T_1 .. S_n \rightarrow T_n] \leq S \rightarrow T$ and $\Gamma \vdash U \leq S$,
then $\Gamma \vdash \bigwedge^*[T_j \mid \Gamma \vdash U \leq S_j] \leq T$.

PROOF: There are two cases to be considered according to the normal form of $S \rightarrow T$. The case when $(S \rightarrow T)^{nf} \equiv S^{nf} \rightarrow T^{nf}$ is similar to but simpler than the one we consider here. Assume

$$(S \rightarrow T)^{nf} \equiv \bigwedge^* [S^{nf} \rightarrow A_1 \dots S^{nf} \rightarrow A_m], \text{ where } T^{nf} \equiv \bigwedge^* [A_1 \dots A_m].$$

By the equivalence of ordinary and normal subtyping (theorem 3.68),

$$\Gamma^{nf} \vdash_n \bigwedge^* [S_1^{nf} \rightarrow B_1^1 \dots S_1^{nf} \rightarrow B_1^{k_1} \dots S_n^{nf} \rightarrow B_n^1 \dots S_n^{nf} \rightarrow B_n^{k_n}] \leq \bigwedge^* [S^{nf} \rightarrow A_1 \dots S^{nf} \rightarrow A_m],$$

$$\text{where } T_i^{nf} = \begin{cases} B_i^1, & \text{if it is not an intersection;} \\ \bigwedge^* [B_i^1 \dots B_i^{k_i}], & \text{otherwise.} \end{cases}$$

By generation for normal subtyping (proposition 3.58), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ and $l_j \in \{1..k_j\}$ such that $\Gamma^{nf} \vdash_n S_j^{nf} \rightarrow B_j^{l_j} \leq S^{nf} \rightarrow A_i$. Again, by generation for normal subtyping (proposition 3.58), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ and $l_j \in \{1..k_j\}$ such that $\Gamma^{nf} \vdash_n S^{nf} \leq S_j^{nf}$ and $\Gamma^{nf} \vdash_n B_j^{l_j} \leq A_i$. By NS-TRANS and the equivalence of ordinary and normal subtyping (theorem 3.68), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ and $l_j \in \{1..k_j\}$ such that $\Gamma \vdash U \leq S_j$ and $\Gamma^{nf} \vdash_n B_j^{l_j} \leq A_i$. By NS- \exists , for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ such that $\Gamma \vdash U \leq S_j$ and $\Gamma^{nf} \vdash_n \bigwedge^* [B_j^1 \dots B_j^{k_j}] \leq A_i$. By the equivalence of ordinary and normal subtyping (theorem 3.68), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ such that $\Gamma \vdash U \leq S_j$ and $\Gamma \vdash T_j \leq A_i$. By lemma 3.28, S-CONV, and S-TRANS, $\Gamma \vdash \bigwedge^* [T_j \mid \Gamma \vdash U \leq S_j] \leq T$. \square

LEMMA 3.87 (*Substitution for subtyping*)

If $\Gamma_1 \vdash S_1 \leq T_1$ and $\Gamma_1, X \leq T_1 : K_1, \Gamma_2 \vdash S_2 \leq T_2$, then $\Gamma_1, \Gamma_2[X \leftarrow S_1] \vdash S_2[X \leftarrow S_1] \leq T_2[X \leftarrow S_1]$.

PROOF: By straightforward induction on the derivation of $\Gamma_1, X \leq T_1 : K_1, \Gamma_2 \vdash S_2 \leq T_2$, using weakening (corollary 3.21), the type substitution lemma (lemma 3.25), and lemma 3.14. \square

LEMMA 3.88 (*Type application*)

If $\Gamma \vdash \bigwedge^* [\forall X \leq S_1 : K_1. T_1 \dots \forall X \leq S_n : K_n. T_n] \leq \forall X \leq S : K. T$ and $\Gamma \vdash U \leq S$, then $\Gamma \vdash \bigwedge^* [T_j[X \leftarrow U] \mid \Gamma \vdash U \leq S_j] \leq T[X \leftarrow U]$.

PROOF: There are two cases to be considered according to the normal form of $\forall X \leq S : K. T$. The case when $(\forall X \leq S : K. T)^{nf} \equiv \forall X \leq S^{nf} : K. T^{nf}$ is similar to but simpler than the one we consider here. Assume $(\forall X \leq S : K. T)^{nf} \equiv \bigwedge^* [\forall X \leq S^{nf} : K. A_1 \dots \forall X \leq S^{nf} : K. A_m]$ where $T^{nf} \equiv \bigwedge^* [A_1 \dots A_m]$. By the equivalence of ordinary and normal subtyping (theorem 3.68),

$$\begin{aligned} \Gamma^{nf} \vdash_n \bigwedge^* [\forall X \leq S_1^{nf} : K_1. B_1^1 \dots \forall X \leq S_1^{nf} : K_1. B_1^{k_1} \dots \forall X \leq S_n^{nf} : K_n. B_n^1 \dots \forall X \leq S_n^{nf} : K_n. B_n^{k_n}] \\ \leq \bigwedge^* [\forall X \leq S^{nf} : K. A_1 \dots \forall X \leq S^{nf} : K. A_m], \end{aligned}$$

$$\text{where } T_i^{nf} = \begin{cases} B_i^1, & \text{if it is not an intersection;} \\ \bigwedge^* [B_i^1 \dots B_i^{k_i}], & \text{otherwise.} \end{cases}$$

By generation for normal subtyping (proposition 3.58), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ and $l_j \in \{1..k_j\}$ such that $\Gamma^{nf} \vdash_n \forall X \leq S_j^{nf} : K_j. B_j^{l_j} \leq \forall X \leq S^{nf} : K. A_i$. Again, by generation for normal subtyping (proposition 3.58), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ and $l_j \in \{1..k_j\}$ such that $K \equiv K_j$, $S^{nf} \equiv S_j^{nf}$, and $\Gamma^{nf}, X \leq S_j^{nf} : K \vdash_n B_j^{l_j} \leq A_i$. By NS- $\forall\exists$, for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ such that $\Gamma^{nf}, X \leq S_j^{nf} : K \vdash_n T_j^{nf} \leq A_i$. By the equivalence of ordinary and normal subtyping (theorem 3.68), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ such that $\Gamma, X \leq S_j : K \vdash T_j \leq A_i$. Furthermore, by S-CONV and S-TRANS, $\Gamma \vdash U \leq S_j$. Then, by the substitution lemma for subtyping (lemma 3.87), for each $i \in \{1..m\}$ there exist $j \in \{1..n\}$ such that $\Gamma \vdash T_j[X \leftarrow U] \leq A_i[X \leftarrow U]$. By NS- $\forall\exists$, $\Gamma \vdash \bigwedge^* [T_j[X \leftarrow U] \mid \Gamma \vdash U \leq S_j] \leq \bigwedge^* [A_1[X \leftarrow U] \dots A_m[X \leftarrow U]]$. By the definition of substitution, $\Gamma \vdash \bigwedge^* [T_j[X \leftarrow U] \mid \Gamma \vdash U \leq S_j] \leq T^{nf}[X \leftarrow U]$. Finally, by lemma 3.14, S-CONV, and S-TRANS, $\Gamma \vdash \bigwedge^* [T_j[X \leftarrow U] \mid \Gamma \vdash U \leq S_j] \leq T[X \leftarrow U]$. \square

Usually, the next step to prove the accuracy of an algorithm, *inf* in our case, would be to prove a completeness result: if the term e has type T with respect to the context Γ in the typing system F_\wedge^ω then the algorithm *inf* finds a type T' for e in Γ . In the present situation this result is not strong enough, since every closed term is typeable in both systems. One easily proves that

LEMMA 3.89

1. If e is closed in Γ , then there exists T such that $\Gamma \vdash e : T$.
2. If e is closed in Γ , then there exists T such that $\Gamma \vdash_{inf} e : T$.

We use the fact that *inf* is deterministic, which means that the rules are invertible, to prove that *inf* finds a minimal type.

PROPOSITION 3.90 (*Generation for inf*) The form of a derivable typing judgement uniquely determines the last applied rule.

PROPOSITION 3.91 (*inf computes minimal types*)

If $\Gamma \vdash e : T$ and $\Gamma \vdash_{inf} e : T'$, then $\Gamma \vdash T' \leq T$.

PROOF: By induction on the derivation of $\Gamma \vdash e : T$. We illustrate the proof technique showing the case for T-APP. We are given that $e \equiv f a$, $\Gamma \vdash f : V \rightarrow T$, and $\Gamma \vdash a : V$. By generation for *inf* (proposition 3.90), $\Gamma \vdash_{inf} f : U$, $\Gamma \vdash_{inf} a : S$, and $T' \equiv \bigwedge^* [T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(U))]$ and $\Gamma \vdash S \leq S_i$. By the induction hypothesis, $\Gamma \vdash U \leq V \rightarrow T$, and $\Gamma \vdash S \leq V$. By lemma 3.84(1), $\Gamma \vdash \bigwedge^* [S_i \rightarrow T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(U))] \leq V \rightarrow T$. Finally, by the term application lemma (lemma 3.86), it follows that $\Gamma \vdash \bigwedge^* [T_i \mid \Gamma \vdash S \leq S_i] \leq T$, where $S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(U))$. In other words, $\Gamma \vdash \bigwedge^* [T_i \mid S_i \rightarrow T_i \in \text{arrows}(\text{flub}_\Gamma(U))] \leq T$. \square

Finally, we have proved the following result.

THEOREM 3.92 (*Minimal typing for F_\wedge^ω*) Given a term e and a context Γ , there exists T such that for every T' , if $\Gamma \vdash e : T'$, then $\Gamma \vdash T \leq T'$.

3.10 Subject reduction

The F_\wedge^ω system is layered in three syntactic categories: kinds, types, and terms. Since terms do not appear in either types or kinds, reductions in type expressions can be studied independently from the reductions of terms. In section 2, we proved that reduction on types preserves kinding properties: the sub-language of types and kinds satisfies the subject reduction property (lemma 3.26):

if $\Gamma \vdash S : K$ and $S \rightarrow_{\beta_\wedge} T$, then $\Gamma \vdash T : K$.

In this section, we show the subject reduction property for typing judgements (proposition 3.99):

if $\Gamma \vdash e : T$ and $e \rightarrow_{\beta_{for}} e'$, then $\Gamma \vdash e' : T$.

In other words, reductions on terms are also safe.

LEMMA 3.93 If $Y \notin \text{FV}(S)$, then

1. $e[Y \leftarrow T][X \leftarrow S] \equiv e[X \leftarrow S][Y \leftarrow T[X \leftarrow S]]$
2. $U[Y \leftarrow T][X \leftarrow S] \equiv U[X \leftarrow S][Y \leftarrow T[X \leftarrow S]]$

PROOF: By induction on the structure of e and U respectively. \square

LEMMA 3.94 (*Substitution for typing*)

1. If $\Gamma_1 \vdash e_1 : S_1$ and $\Gamma_1, x:S_1, \Gamma_2 \vdash e_2 : S_2$, then $\Gamma_1, \Gamma_2 \vdash e_2[x \leftarrow e_1] : S_2$.
2. If $\Gamma_1 \vdash S \leq S_1$ and $\Gamma_1, X \leq S_1 : K_1, \Gamma_2 \vdash e_2 : S_2$, then $\Gamma_1, \Gamma_2[X \leftarrow S] \vdash e_2[X \leftarrow S] : S_2[X \leftarrow S]$.

PROOF:

1. By induction on the derivation of $\Gamma_1, x:S_1, \Gamma_2 \vdash e_2 : S_2$.

2. By induction on the derivation of $\Gamma_1, X \leq S_1 : K_1, \Gamma_2 \vdash e_2 : S_2$, using the type substitution lemma (lemma 3.25) in the T-VAR and T-MEET cases; the substitution lemma for subtyping (lemma 3.87) and lemma 3.93 in the case for T-TAPP; lemma 3.93 in the T-FOR case, and the substitution lemma for subtyping (lemma 3.87) in the T-SUBSUMPTION case. \square

LEMMA 3.95 $\Gamma \vdash \top^* \leq T$ if and only if $T =_{\beta\wedge} \top^*$ and $\Gamma \vdash T : \star$.

PROOF: If $T =_{\beta\wedge} \top^*$, then the result follows by S-CONV. Otherwise, if $\Gamma \vdash \top^* \leq T$, by the well-kindedness of subtyping (proposition 3.32), T-MEET, and uniqueness of kinds (lemma 3.24), $\Gamma \vdash T : \star$. By the equivalence of ordinary and algorithmic subtyping (proposition 3.74), $\Gamma^{nf} \vdash_{Alg} \top^* \leq T^{nf}$, which can only be derived using ALGS- $\forall\exists$ where T^{nf} is the empty intersection. \square

Given $\Gamma \vdash S \leq T$, generation for normal subtyping (proposition 3.58) and the equivalence of ordinary and normal subtyping (theorem 3.68) provide subtyping information about the normal forms of S and T . We can also show that subtyping is structural for arrow types, quantified types and type operators, without reducing the terms in the subtyping relation to normal form. An implementation of a subtyping algorithm for F_{\wedge}^{ω} could take advantage of this fact by delaying normalizing steps, which might result in having to consider fewer recursive calls or calls with smaller arguments.

LEMMA 3.96 (*Generation for ordinary subtyping*) Let $S_2 \neq_{\beta\wedge} \top^*$. Then:

1. $\Gamma \vdash T_1 \rightarrow T_2 \leq S_1 \rightarrow S_2$ if and only if $\Gamma \vdash S_1 \leq T_1$ and $\Gamma \vdash T_2 \leq S_2$.
2. $\Gamma \vdash \forall X \leq T_1 : K_T . T_2 \leq \forall X \leq S_1 : K_S . S_2$ if and only if $K_S \equiv K_T$, $T_1 =_{\beta\wedge} S_1$, and $\Gamma, X \leq T_1 : K_T \vdash T_2 \leq S_2$.
3. $\Gamma \vdash \Lambda X : K_T . T_2 \leq \Lambda X : K_S . S_2$ if and only if $\Gamma, X : K_S \vdash T_2 \leq S_2$ and $K_T \equiv K_S$.

PROOF: The three statements are proved using a similar argument. We consider here the proof of part 2. If $K_S \equiv K_T$, $T_1 =_{\beta\wedge} S_1$, and $\Gamma, X \leq T_1 : K_T \vdash T_2 \leq S_2$, then, by S-ALL and S-CONV, $\Gamma \vdash \forall X \leq T_1 : K_T . T_2 \leq \forall X \leq S_1 : K_S . S_2$. Conversely, let $\Gamma \vdash \forall X \leq T_1 : K_T . T_2 \leq \forall X \leq S_1 : K_S . S_2$ and $S_2 \neq_{\beta\wedge} \top^*$. Lemma 3.95 implies that $T_2^{nf} \neq_{\beta\wedge} \top^*$. Then we have to consider four cases according to whether S_2^{nf} and T_2^{nf} are intersection types or not. We illustrate the proof argument considering just one case. Let

$$\begin{aligned} (\forall X \leq T_1 : K_T . T_2)^{nf} &\equiv \forall X \leq T_1^{nf} : K_T . T_2^{nf}, \quad \text{and} \\ (\forall X \leq S_1 : K_S . S_2)^{nf} &\equiv \bigwedge^* [\forall X \leq S_1^{nf} : K_S . A_1 .. \forall X \leq S_1^{nf} : K_S . A_n], \end{aligned}$$

where $S_2^{nf} \equiv \bigwedge^* [A_1 .. A_n]$. By the equivalence of ordinary and normal subtyping (theorem 3.68) and generation for normal subtyping (proposition 3.58), for each $i \in \{1..n\}$ $\Gamma^{nf} \vdash_n \forall X \leq T_1^{nf} : K_T . T_2^{nf} \leq \forall X \leq S_1^{nf} : K_S . A_i$ and, again generation for normal subtyping (proposition 3.58) implies that $\Gamma^{nf}, X \leq T_1^{nf} : K_T \vdash_n T_2^{nf} \leq A_i$, and $T_1^{nf} \equiv S_1^{nf}$. By NS- \forall , $\Gamma^{nf}, X \leq T_1^{nf} : K_T \vdash_n T_2^{nf} \leq S_2^{nf}$ and $\Gamma, X \leq T_1 : K_T \vdash T_2 \leq S_2$, by the equivalence of ordinary and normal subtyping (theorem 3.68). \square

LEMMA 3.97 (*Generation for typing*)

1. If $\Gamma \vdash \lambda x : S_1 . e : S$, then there exists S_2 such that $\Gamma, x : S_1 \vdash e : S_2$ and $\Gamma \vdash S_1 \rightarrow S_2 \leq S$.
2. If $\Gamma \vdash \lambda X \leq S_1 : K_1 . e : S$, then there exists S_2 such that $\Gamma, X \leq S_1 : K_1 \vdash e : S_2$ and $\Gamma \vdash \forall X \leq S_1 : K_1 . S_2 \leq S$.
3. If $\Gamma \vdash \text{for}(X \in \{U_1 .. U_n\}) e : T$, then there exist $T_1 .. T_n$ such that, for each i in $\{1..n\}$, $\Gamma \vdash e[X \leftarrow U_i] : T_i$ and $\Gamma \vdash \bigwedge^* [T_1 .. T_n] \leq T$.

PROOF: Each statement is proved by induction on the derivation of the typing judgement in the antecedent. We exhibit here the proof of part 3. We proceed by case analysis on the last rule of the derivation of $\Gamma \vdash \text{for}(X \in \{U_1 .. U_n\}) e : T$.

T-FOR We are given that $\Gamma \vdash e[X \leftarrow U] : T$ for some $U \in \{U_1..U_n\}$. Since every closed term has a type, we have that, for each i in $\{1..n\}$, $\Gamma \vdash e[X \leftarrow U_i] : T_i$, and, by **S-MEET-LB**, $\Gamma \vdash \bigwedge^*[T_1..T..T_n] \leq T$.

T-MEET Let $T \equiv \bigwedge^*[S_1..S_k]$. We are given that, $\Gamma \vdash \text{ok}$ and $\Gamma \vdash \text{for}(X \in \{U_1..U_n\})e : S_j$, for each j in $\{1..k\}$. By the induction hypothesis, for each $j \in \{1..k\}$ and each $i \in \{1..n\}$, there exist T_{j_i} such that $\Gamma \vdash e[X \leftarrow U_i] : T_{j_i}$, and $\Gamma \vdash \bigwedge^*[T_{j_1}..T_{j_n}] \leq S_j$, and, by the minimal type property (theorem 3.92), there exist $T_1..T_n$ such that $\Gamma \vdash e[X \leftarrow U_i] : T_i$, and $\Gamma \vdash T_i \leq T_{j_i}$, by lemma 3.28, it follows that $\Gamma \vdash \bigwedge^*[T_1..T_n] \leq \bigwedge^*[T_{j_1}..T_{j_n}]$, and by **S-TRANS**, $\Gamma \vdash \bigwedge^*[T_1..T_n] \leq S_j$. Finally, by **S-MEET-G**, it follows that $\Gamma \vdash \bigwedge^*[T_1..T_n] \leq \bigwedge^*[S_1..S_k]$.

T-SUB We are given that $\Gamma \vdash \text{for}(X \in \{U_1..U_n\})e : S$, and $\Gamma \vdash S \leq T$. The result follows by the induction hypothesis and **S-TRANS**. \square

Since terms cannot occur in types, subject reduction for terms does not need to consider reductions in contexts.

PROPOSITION 3.98 (*One step subject reduction for typing judgements*)

If $\Gamma \vdash e : T$ and $e \rightarrow_{\beta \text{for}} e'$, then $\Gamma \vdash e' : T$.

PROOF: Since every term has type \top^* , the interesting case is when $T \neq_{\beta \wedge} \top^*$. This proposition follows by induction on the derivation of $\Gamma \vdash e : T$. We consider the cases where e is a redex; the other cases follow by straightforward application of the induction hypothesis.

T-APP There are two possibilities for e to be a redex.

1. $e \equiv (\lambda x:S_1.e_1) e_2$, $e' \equiv e_1[x \leftarrow e_2]$, and $T \equiv T_2$. We are given that $\Gamma \vdash \lambda x:S_1.e_1 : T_1 \rightarrow T_2$ and $\Gamma \vdash e_2 : T_1$. By the generation lemma for typing (lemma 3.97), there exists S_2 such that, $\Gamma, x:S_1 \vdash e_1 : S_2$ and $\Gamma \vdash S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2$. Since $T_2 \neq_{\beta \wedge} \top^*$, by the generation lemma for ordinary subtyping (lemma 3.96), $\Gamma \vdash T_1 \leq S_1$ and $\Gamma \vdash S_2 \leq T_2$. Then, by **T-SUBSUMPTION**, it follows that $\Gamma, x:S_1 \vdash e_1 : T_2$ and $\Gamma \vdash e_2 : S_1$. Finally, by the substitution lemma for typing (lemma 3.94(1)), $\Gamma \vdash e_1[x \leftarrow e_2] : T_2$.
2. $e \equiv (\text{for}(X \in U_1..U_n)e_2) e_1$, $e' \equiv \text{for}(X \in U_1..U_n)(e_2 e_1)$, and $T \equiv T_2$. We are given that $\Gamma \vdash \text{for}(X \in U_1..U_n)e_2 : T_1 \rightarrow T_2$ and $\Gamma \vdash e_1 : T_1$. By the generation lemma for typing (lemma 3.97), there exist $V_1..V_n$ such that $\Gamma \vdash e_2[X \leftarrow U_i] : V_i$ for each $i \in \{1..n\}$, and $\Gamma \vdash \bigwedge^*[V_1..V_n] \leq T_1 \rightarrow T_2$.

We write $V_i^{nf} \equiv A_{i_1}$, if it is not an intersection,
 $V_i^{nf} \equiv \bigwedge^*[A_{i_1}..A_{i_{k_i}}]$, otherwise.

Note that $\bigwedge^*[V_1..V_n]^{nf} \equiv \bigwedge^*[A_{1_1}..A_{1_{k_1}}..A_{n_1}..A_{n_{k_n}}]$. By the equivalence of ordinary and normal subtyping (theorem 3.68), $\Gamma^{nf} \vdash_n \bigwedge^*[A_{1_1}..A_{1_{k_1}}..A_{n_1}..A_{n_{k_n}}] \leq (T_1 \rightarrow T_2)^{nf}$. There are two cases to consider according to the form of $(T_1 \rightarrow T_2)^{nf}$. If $(T_1 \rightarrow T_2)^{nf}$ is $T_1^{nf} \rightarrow T_2^{nf}$ or $\bigwedge^*[T_1^{nf} \rightarrow B_1..T_1^{nf} \rightarrow B_r]$ where $T_2^{nf} \equiv \bigwedge^*[B_1..B_r]$. We show here the latter; the former is simpler.

By generation for normal subtyping (proposition 3.58), for every $s \in \{1..r\}$ there exist $l \in \{1..n\}$ and $j \in \{1..k_l\}$ such that $\Gamma^{nf} \vdash_n A_{l_j} \leq T_1^{nf} \rightarrow B_s$, and, by **NS- \exists** or **NS-REFL**, $\Gamma^{nf} \vdash_n V_l^{nf} \leq A_{l_j}$, by **NS-TRANS**, for every $s \in \{1..r\}$ there exists $l \in \{1..n\}$ such that $\Gamma^{nf} \vdash_n V_l^{nf} \leq T_1^{nf} \rightarrow B_s$, and, by the equivalence of ordinary and normal subtyping (theorem 3.68), $\Gamma \vdash V_l \leq T_1 \rightarrow B_s$. By **T-SUBSUMPTION**, for every $s \in \{1..r\}$ there exists $l \in \{1..n\}$ $\Gamma \vdash e_2[X \leftarrow S_l] : T_1 \rightarrow B_s$. By **T-APP**, for every $s \in \{1..r\}$ there exists $l \in \{1..n\}$ $\Gamma \vdash (e_2[X \leftarrow S_l]) e_1 : B_s$, and since X is not a free variable of e_1 we have that, for every $s \in \{1..r\}$ there exists $l \in \{1..n\}$ $\Gamma \vdash e_2 e_1[X \leftarrow S_l] : B_s$. Applying **T-FOR**, we get that for every $s \in \{1..r\}$ $\Gamma \vdash \text{for}(X \in U_1..U_n)e_2 e_1 : B_s$, by **T-MEET**, $\Gamma \vdash \text{for}(X \in U_1..U_n)e_2 e_1 : T_2^{nf}$. Finally, by **S-CONV** and **T-SUBSUMPTION**, $\Gamma \vdash \text{for}(X \in U_1..U_n)e_2 e_1 : T_2$.

T-TAPP There are two possibilities for e to be a redex. We show only one case here. The case when $e \equiv (\text{for}(X \in U_1..U_n)e_2) S$ follows a similar argument to the one used for the case $e \equiv (\text{for}(X \in U_1..U_n)e_2) e_1$ in **T-APP**.

If $e \equiv (\lambda X \leq S_1 : K_S.e_2) S$, $e' \equiv e_2[X \leftarrow S]$, and $T \equiv T_2[X \leftarrow S]$, we have that $\Gamma \vdash \lambda X \leq S_1 : K_S.e_2 : \forall X \leq T_1 : K_T.T_2$ and $\Gamma \vdash S \leq T_1$. By the generation lemma for typing (lemma 3.97), there exists S_2 such that $\Gamma, X \leq S_1 : K_S \vdash e_2 : S_2$ and $\Gamma \vdash \forall X \leq S_1 : K_S.S_2 \leq \forall X \leq T_1 : K_T.T_2$. Since $T_2[X \leftarrow S] \neq_{\beta\wedge} \top^*$, lemma 3.14 implies that $T_2 \neq_{\beta\wedge} \top^*$. Then, by the generation lemma for ordinary subtyping (lemma 3.96), $\Gamma, X \leq S_1 : K_S \vdash S_2 \leq T_2$, $S_1 =_{\beta\wedge} T_1$, and $K_S \equiv K_T$. By **T-SUBSUMPTION**, $\Gamma, X \leq S_1 : K_S \vdash e_2 : T_2$, and, by **S-TRANS** and **S-CONV**, $\Gamma \vdash S \leq S_1$. Finally, by the substitution lemma for typing (lemma 3.94(2)), $\Gamma \vdash e_2[X \leftarrow S] : T_2[X \leftarrow S]$.

T-FOR Let $e \equiv \text{for}(X \in U_1..U_n)e_1$, where $X \notin \text{FTV}(e_1)$ and $e' \equiv e_1$. We are given that $\Gamma \vdash e_1[X \leftarrow U] : T$, with $U \in \{U_1..U_n\}$. Since $e_1 \equiv e_1[X \leftarrow U]$, the result holds. \square

We now have all the results needed in order to prove that reduction on terms preserves typing. The following proposition, the subject reduction property for F_λ^ω terms, is a consequence of the previous one.

PROPOSITION 3.99 (*Subject reduction for typing judgements*)

If $\Gamma \vdash e : T$ and $e \rightarrow_{\beta\text{for}} e'$, then $\Gamma \vdash e' : T$.

PROOF: By induction on the derivation of $e \rightarrow_{\beta\text{for}} e'$, using proposition 3.98. \square

4 Decidability

4.1 Decidability of context formation and kinding

The decidability of kinding is used to prove the decidability of subtyping in section 4.2.1, and the decidability of ok judgements is used to prove the decidability of typechecking in section 4.4. We want to define a measure (size) on judgements to show that the complexity of the hypothesis in a kinding derivation rule is smaller than the complexity of the conclusion. Given that kinding rules may depend on ok judgements we need to define a measure for both kinding and well-formation judgements. Rules **C-TVAR** and **C-VAR** suggest that the size of ok should be bigger than 0 (let it be 1). The rule **K-MEET** in the empty case, suggests that the size of \top^K should be bigger than the size of ok, while **K-TVAR** indicates that the size of a variable should be bigger than the size of ok. With all these ideas in mind we define the following measure.

DEFINITION 4.1 (*Size*)

1. The size of a type expression T , $size_t(T)$, is defined as follows.
 - (a) $size_t(X) = 2$,
 - (b) $size_t(S \rightarrow T) = size_t(\forall X \leq S : K.T) = size_t(ST) = size_t(S) + size_t(T) + 1$,
 - (c) $size_t(\lambda X : K.T) = size_t(T) + 3$,
 - (d) $size_t(\bigwedge^K [T_1..T_n]) = 2 + \sum_{1 \leq i \leq n} size_t(T_i)$.
2. The homomorphic extension to contexts, $size_c(\Gamma)$, is defined as follows.
 - (a) $size_c(\emptyset) = 0$,
 - (b) $size_c(\Gamma, X \leq T : K) = size_c(\Gamma, x : T) = size_c(\Gamma) + size_t(T)$.
3. The size of a subtyping, kinding, or ok judgement J , $size_j(J)$, is defined as follows.
 - (a) $size_j(\Gamma \vdash \text{ok}) = size_c(\Gamma) + 1$,
 - (b) $size_j(\Gamma \vdash T : K) = size_c(\Gamma) + size_t(T)$.

$$(c) \text{ size}_j(\Gamma \vdash S \leq T) = \text{size}_c(\Gamma) + \text{size}_t(S) + \text{size}_t(T).$$

LEMMA 4.2 (*Well-foundedness of context formation and kinding rules*)

1. For every kinding or ok judgement J , $\text{size}_j(\emptyset \vdash \text{ok}) \leq \text{size}_j(J)$.
2. If $\frac{J_1 \dots J_n}{J}$ is a kinding rule or a context formation rule, then $\text{size}_j(J_i) < \text{size}_j(J)$ for each $i \in \{1..n\}$.

COROLLARY 4.3 (*Decidability of context formation and kinding*)

1. For any context Γ it is decidable whether $\Gamma \vdash \text{ok}$.
2. For any context Γ , type expression T , and kind K , it is decidable whether $\Gamma \vdash T : K$.

PROOF: Lemma 3.16 and proposition 3.23 imply that context formation rules and kinding rules determine an algorithm to check context judgements and kinding judgements, and lemma 4.2 implies that this algorithm terminates. \square

4.2 Decidability of subtyping

In the solution for the second order lambda calculus presented in [47], the distributivity rules for intersection types are not considered as rewrite rules. For that reason, new syntactic categories have to be defined (composite and individual canonical types) and an auxiliary mapping (flattening) transforms a type into a canonical type. Our solution does not need either new syntactic categories or elaborate auxiliary mappings, since the role played there by canonical types is performed here by types in normal form.

After the work in [25] was completed, Steffen and Pierce proved a similar result for F_{\leq}^{ω} [48]. There are several differences between our work and the proof of decidability of subtyping in [48]. Our result is for a stronger system which also includes intersection types, and implies the result for F_{\leq}^{ω} . Moreover, our proof of termination has the novel idea of using a choice operator to model the behavior of type variables during subtype checking.

An important property of derivation systems is the information that a derivable judgement contains about its proofs. This information is essential to produce results which not only state properties about the subproofs, but also help identify ill-formed judgements. Example 3.44 illustrates how the bounds of type variables are used in subtyping derivations showing that the types considered in the premises of S-TRANS can be larger (in number of symbols) than those in their conclusion.

4.2.1 Termination of subtype checking

Our proof of termination was inspired by the observation that not all occurrences of the same type variable are replaced by the bound of the variable during subtype checking. That led us to define a mapping (*plus*) on types where each type variable is enriched with this non-deterministic behavior (each variable is expanded with a choice containing itself and its bound, and recursively applied to the bound). Furthermore, the bound of a variable has fewer options in this non-deterministic behavior (its expansion has fewer choices), suggesting a decreasing measure to prove that $\text{Alg}F_{\lambda}^{\omega}$ is well-founded.

To establish the decidability of the subtyping relation of F_{λ}^{ω} we prove the well-foundedness of the relation defined by the $\text{Alg}F_{\lambda}^{\omega}$ subtyping rules. We show this by reducing the well-foundedness of $\text{Alg}F_{\lambda}^{\omega}$ to the strong normalization property of the $\rightarrow_{\beta\wedge+}$ relation defined below, and we use the well-foundedness of $\text{Alg}F_{\lambda}^{\omega}$ to show that checking whether a given subtyping judgement is derivable always terminates.

We begin by extending the language of types with the constructor $+$ as follows.

\mathbb{T}^+	$::=$	X	type variable
		$ \ \mathbb{T}^+ \rightarrow \mathbb{T}^+$	function type
		$ \ \forall(X \leq \mathbb{T}^+ : \mathbb{K}) \mathbb{T}^+$	polymorphic type
		$ \ \Lambda(X : \mathbb{K}) \mathbb{T}^+$	operator abstraction
		$ \ \mathbb{T}^+ \mathbb{T}^+$	operator application
		$ \ \bigwedge^{\mathbb{K}} [\mathbb{T}^+ .. \mathbb{T}^+]$	intersection at kind \mathbb{K}
		$ \ \mathbb{T}^+ + \mathbb{T}^+$	choice

Since we have enriched the language of types with a new type constructor, we need to extend our kinding judgements (section 2.4) with the following kinding rule.

$$\frac{\Gamma \vdash_+ S : K \quad \Gamma \vdash_+ T : K}{\Gamma \vdash_+ S + T : K} \quad (\text{K-PLUS})$$

The reduction $\rightarrow_{\beta\wedge+}$ is obtained from $\rightarrow_{\beta\wedge}$ by adding the reductions associated with the choice operator $+$, $S + T \rightarrow_{\beta\wedge+} S$ and $S + T \rightarrow_{\beta\wedge+} T$. We also need the corresponding kinding rule saying that $\Gamma \vdash S + T : K$ whenever $\Gamma \vdash S, T : K$. As far as we are aware, choice operators have not been used before to analyze subtyping.

NOTATION 4.4 We write $+$ modulo commutativity and associativity.

We now define a new reduction $\rightarrow_{\beta\wedge+}$.

DEFINITION 4.5 ($\rightarrow_{\beta\wedge+}$) The reduction on types $\rightarrow_{\beta\wedge+}$ is obtained from $\rightarrow_{\beta\wedge}$ (definition 2.1) by adding the following two rules:

1. $S + T \rightarrow_{\beta\wedge+} S$, and
2. $S + T \rightarrow_{\beta\wedge+} T$.

We also write \rightarrow_+ to refer to these two new reduction rules.

As usual, $\rightarrow_{\beta\wedge+}$ is extended to become a compatible relation with respect to type formation, $\twoheadrightarrow_{\beta\wedge+}$ is the reflexive, transitive closure of $\rightarrow_{\beta\wedge+}$, and $=_{\beta\wedge+}$ is the reflexive, symmetric, and transitive closure of $\rightarrow_{\beta\wedge+}$.

PROPOSITION 4.6 (*Strong normalization for $\rightarrow_{\beta\wedge+}$*)

If $\Gamma \vdash_+ T : K$, then every $\beta\wedge+$ -reduction sequence starting from T is finite.

PROOF: The result follows using the strategy used to prove that the reduction $\rightarrow_{\beta\wedge}$ is strongly normalizing on well kinded types (see section 3.3). We only need to modify the definition of saturated sets by adding the following closure condition:

if $T, U, R_1 .. R_n \in SN^+$, then $TR_1 .. R_n \in S$ and $UR_1 .. R_n \in S$ imply $(T + U)R_1 .. R_n \in S$. \square

Next, we define a measure for subtyping judgements such that, given a subtyping rule, the measure of each hypothesis is smaller than that of the conclusion. Most measures for showing the well-foundedness of a relation defined by a set of inference rules involve a clever assignment of weights to judgements, often involving the number of symbols. We need a more complex measure, since in ALGS-OAPP it is not necessarily the case that the size of the hypothesis is smaller than the size of the conclusion.

We introduce a new mapping from types to types in the extended language in order to define a new measure on subtyping judgements. To motivate the definition of this new measure, we analyze the behavior of type variables during subtype checking. Assume that we want to check if $\Gamma \vdash_{Alg} S \leq T$, where S is a variable or a type application. It can be the case that the judgement is obtained with an application of ALGS-TVAR or ALGS-OAPP, in which case we have to consider a new judgement $\Gamma \vdash_{Alg} S' \leq T$, where S' is obtained from S by replacing a variable by its bound (and eventually normalizing). However, we do not replace every variable by its bound, as this would constitute an unsound operation with respect to subtyping. Consider the following example.

EXAMPLE 4.7 Two unrelated variables may have the same bound.

$$\begin{aligned} X \leq \top^* : \star, Y \leq \top^* : \star \not\vdash X \leq Y, \quad \text{but} \\ X \leq \top^* : \star, Y \leq \top^* : \star \vdash \top^* \leq \top^*. \end{aligned}$$

Our new mapping, *plus*, includes in each type expression this nondeterministic behavior of its type variables.

DEFINITION 4.8 (*plus*)

The mapping $plus_\Gamma : \mathbb{T} \rightarrow \mathbb{T}^+$ is defined as follows.

1. $plus_{\Gamma_1, X \leq T : K, \Gamma_2}(X) = X + plus_{\Gamma_1}(T)$,
2. $plus_\Gamma(T \rightarrow S) = plus_\Gamma(T) \rightarrow plus_\Gamma(S)$,
3. $plus_\Gamma(\forall X \leq T : K. S) = \forall X \leq plus_\Gamma(T) : K. plus_{\Gamma, X \leq T : K}(S)$,
4. $plus_\Gamma(\Lambda X : K. S) = \Lambda X : K. plus_{\Gamma, X : K}(S)$,
5. $plus_\Gamma(ST) = plus_\Gamma(S) plus_\Gamma(T)$,
6. $plus_\Gamma(\bigwedge^K [S_1 \dots S_n]) = \bigwedge^K [plus_\Gamma(S_1) \dots plus_\Gamma(S_n)]$.

EXAMPLE 4.9 $plus_{X \leq \top^* : \star, Y \leq X : \star, Z \leq Y : \star}(Z) = Z + Y + X + \top^*$.

We need to show that *plus* is well defined on well kinded arguments.

LEMMA 4.10 (*Well-foundedness of plus*) If $\Gamma \vdash T : K$, then $plus_\Gamma(T)$ is defined.

PROOF: Observe that the $size_j$ of the kinding judgements of the arguments strictly decreases in each recursive call. Consider

$$rank_\Gamma(S) = size_j(\Gamma \vdash S : kind(\Gamma, S)),$$

where $size_j(\Gamma \vdash S : K)$ is the size of the derivation of the kinding judgement (see definition 4.1). The function $kind$ can be defined straightforwardly using proposition 3.23, such that $kind(\Gamma, S) = K$ if $\Gamma \vdash S : K$, and gives a constant *NoKind* otherwise. Moreover, lemma 4.2 implies that the function $kind$ is total. Given that $\Gamma \vdash S : K$, by lemmas 3.16(1) and 3.23, the rank decreases in each recursive call and the least value is that of $size_j(\vdash \top^K : K)$. \square

LEMMA 4.11 If $\Gamma \vdash T : K$, then $\Gamma \vdash_+ plus_\Gamma(T) : K$.

PROOF: By induction on the derivation of $\Gamma \vdash T : K$, observing that $\Gamma \vdash T : K$ implies $\Gamma \vdash_+ T : K$. It is straightforward to verify that \vdash_+ satisfies weakening using renamings (see corollary 3.21). We consider here the case for K-TVAR, the rest follows by straightforward induction. We are given, $\Gamma_1, X \leq T : K, \Gamma_2 \vdash \text{ok}$. By lemma 3.16, there is a proper subderivation of $\Gamma_1 \vdash T : K$. Finally, the result follows by the induction hypothesis, weakening, and K-PLUS. \square

LEMMA 4.12 (*Strengthening for plus*)

1. Let $X \notin \text{FTV}(\Gamma_2) \cup \text{FTV}(S)$. Then $\Gamma_1, X \leq T_X : K_X, \Gamma_2 \vdash S : K$ implies $plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2}(S) = plus_{\Gamma_1, \Gamma_2}(S)$.
2. $\Gamma_1, x : T, \Gamma_2 \vdash S : K$ implies $plus_{\Gamma_1, x : T, \Gamma_2}(S) = plus_{\Gamma_1, \Gamma_2}(S)$.

PROOF:

1. By lemma 4.10, $plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2}(S)$ is defined, therefore we can reason by induction on the number of unfolding steps of *plus*. We proceed by case analysis on the form of S .

$S \equiv Y$. We have to consider two cases.

(a) $\Gamma_1 \equiv \Delta_1, Y \leq T_1 : K_1, \Delta_2$. Then, by definition,

$$plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2}(Y) = Y + plus_{\Delta_1}(T_1).$$

On the other hand, also by the definition of *plus*,

$$plus_{\Gamma_1, \Gamma_2}(Y) = Y + plus_{\Delta_1}(T_1).$$

(b) $\Gamma_2 \equiv \Delta_1, Y \leq T_1 : K_1, \Delta_2$. By the definition of *plus*,

$$plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2}(Y) = Y + plus_{\Gamma_1, X \leq T_X : K_X, \Delta_1}(T_1).$$

By lemma 3.15,

$$\Gamma_1, X \leq T_X : K_X, \Gamma_2 \vdash \text{ok},$$

and, by lemma 3.16(1),

$$\Gamma_1, X \leq T_X : K_X, \Delta_1 \vdash T_1 : K_1.$$

Moreover, since $X \notin \text{FTV}(\Gamma_2)$, it follows that $X \notin \text{FTV}(\Delta_1) \cup \text{FTV}(T_1)$. Then, applying the induction hypothesis we obtain

$$Y + plus_{\Gamma_1, X \leq T_X : K_X, \Delta_1}(T_1) = Y + plus_{\Gamma_1, \Delta_1}(T_1),$$

and the result follows by the definition of *plus*.

$S \equiv \forall Y \leq T_1 : K_1.T_2$. By the definition of *plus*,

$$\begin{aligned} & plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2}(\forall Y \leq T_1 : K_1.T_2) \\ &= \forall Y \leq plus_{\Gamma_1, \leq T_X : K_X, \Gamma_2}(T_1) : K_1 \cdot plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2, Y \leq T_1 : K_1}(T_2). \end{aligned}$$

By generation for kinding (proposition 3.23),

$$\Gamma_1, X \leq T_X : K_X, \Gamma_2, Y \leq T_1 : K_1 \vdash T_2 : \star,$$

and, since $X \notin \text{FTV}(\Gamma_2, Y \leq T_1 : K_1) \cup \text{FTV}(T_2)$, by the induction hypothesis,

$$\begin{aligned} & \forall Y \leq plus_{\Gamma_1, \leq T_X : K_X, \Gamma_2}(T_1) : K_1 \cdot plus_{\Gamma_1, X \leq T_X : K_X, \Gamma_2, Y \leq T_1 : K_1}(T_2) \\ &= \forall Y \leq plus_{\Gamma_1, \leq T_X : K_X, \Gamma_2}(T_1) : K_1 \cdot plus_{\Gamma_1, \Gamma_2, Y \leq T_1 : K_1}(T_2). \end{aligned}$$

By lemma 3.15,

$$\Gamma_1, X \leq T_X : K_X, \Gamma_2, Y \leq T_1 : K_1 \vdash \text{ok},$$

by generation for context judgements (lemma 3.16(1)),

$$\Gamma_1, X \leq T_X : K_X, \Gamma_2 \vdash T_1 : K_1.$$

Since $X \notin \text{FTV}(\Gamma_2) \cup \text{FTV}(T_1)$, by the induction hypothesis,

$$\begin{aligned} & \forall Y \leq plus_{\Gamma_1, \leq T_X : K_X, \Gamma_2}(T_1) : K_1 \cdot plus_{\Gamma_1, \Gamma_2, Y \leq T_1 : K_1}(T_2) \\ &= \forall Y \leq plus_{\Gamma_1, \Gamma_2}(T_1) : K_1 \cdot plus_{\Gamma_1, \Gamma_2, Y \leq T_1 : K_1}(T_2) \\ &= plus_{\Gamma_1, \Gamma_2}(\forall Y \leq T_1 : K_1.T_2). \end{aligned}$$

For all the other cases, the result follows by straightforward application of the induction hypothesis, using generation for kinding (proposition 3.23).

2. The definition of *plus* does not depend on the assumptions of term variables. \square

LEMMA 4.13 (*Weakening for plus*) If $\Gamma' \vdash \text{ok}$, $\Gamma \subseteq \Gamma'$, and $\Gamma \vdash S : K$, then $plus_{\Gamma}(S) = plus_{\Gamma'}(S)$.

PROOF: The assumptions ensure that $plus_{\Gamma}(S)$ is defined, so we can proceed by induction on the number of unfolding steps of the definition of *plus*. We proceed by case analysis on the form of S .

$S \equiv X$. By generation for kinding (proposition 3.23) and the fact that $\Gamma \subseteq \Gamma'$,

$$\Gamma \equiv \Gamma_1, X \leq T : K, \Gamma_2 \quad \text{and} \quad \Gamma' \equiv \Gamma'_1, X \leq T : K, \Gamma'_2.$$

There are two cases to consider.

1. If $\Gamma_1 \equiv \Gamma'_1$, then the result follows by the definition of *plus*.
2. If $\Gamma_1 \not\equiv \Gamma'_1$, then $\Gamma_1 \subseteq \Gamma'_1 \cup \Gamma'_2$.

By the definition of *plus*,

$$\text{plus}_{\Gamma}(X) = X + \text{plus}_{\Gamma_1}(T).$$

By lemmas 3.15 and 3.16(1), it follows that $\Gamma_1 \vdash T : K$. Hence, by the induction hypothesis,

$$X + \text{plus}_{\Gamma_1}(T) = X + \text{plus}_{\Gamma'}(T).$$

Since $\Gamma' \vdash \text{ok}$, from lemma 3.16(1), it follows that $\Gamma'_1 \vdash T : K$. Consequently, $(\{X\} \cup \text{FTV}(\Gamma'_2)) \cap \text{FTV}(T) = \emptyset$ by the free variables lemma (lemma 3.17). Hence, starting from the last declaration in Γ'_2 , we can iterate the strengthening lemma for *plus* (lemma 4.12 items 1 and 2) to obtain

$$X + \text{plus}_{\Gamma'}(T) = X + \text{plus}_{\Gamma'_1}(T) = \text{plus}_{\Gamma'}(X).$$

$S \equiv \forall X \leq T_1 : K_1. T_2$. We have that $\Gamma \vdash \forall X \leq T_1 : K_1. T_2 : \star$, by generation for kinding (proposition 3.23).

Let Z be a new type variable, in particular $Z \notin \text{dom}(\Gamma')$. Then, by α -conversion, $S =_{\alpha} \forall Z \leq T_1 : K_1. T_2[X \leftarrow Z]$.

By the definition of *plus*,

$$\begin{aligned} \text{plus}_{\Gamma}(S) &=_{\alpha} \text{plus}_{\Gamma}(\forall Z \leq T_1 : K_1. T_2[X \leftarrow Z]) \\ &= \forall Z \leq \text{plus}_{\Gamma}(T_1) : K_1. \text{plus}_{\Gamma, Z \leq T_1 : K_1}(T_2[X \leftarrow Z]). \end{aligned}$$

By generation for kinding and lemmas 3.15 and 3.16(1), it follows that $\Gamma \vdash T_1 : K_1$. Then, by the induction hypothesis,

$$\begin{aligned} \forall Z \leq \text{plus}_{\Gamma}(T_1) : K_1. \text{plus}_{\Gamma, Z \leq T_1 : K_1}(T_2[X \leftarrow Z]) \\ = \forall Z \leq \text{plus}_{\Gamma'}(T_1) : K_1. \text{plus}_{\Gamma, Z \leq T_1 : K_1}(T_2[X \leftarrow Z]). \end{aligned}$$

By generation for kinding, $\Gamma, X \leq T_1 : K_1 \vdash T_2 : \star$, and by renaming (lemma 3.19), we have that $\Gamma, Z \leq T_1 : K_1 \vdash T_2[X \leftarrow Z] : \star$. (Taking the renaming that maps X to Z and is the identity elsewhere.)

Applying again the induction hypothesis, it follows that

$$\begin{aligned} \forall Z \leq \text{plus}_{\Gamma'}(T_1) : K_1. \text{plus}_{\Gamma, Z \leq T_1 : K_1}(T_2[X \leftarrow Z]) \\ = \forall Z \leq \text{plus}_{\Gamma'}(T_1) : K_1. \text{plus}_{\Gamma', Z \leq T_1 : K_1}(T_2[X \leftarrow Z]) \\ = \text{plus}_{\Gamma'}(\forall Z \leq T_1 : K_1. T_2[X \leftarrow Z]) \\ =_{\alpha} \text{plus}_{\Gamma'}(S). \end{aligned}$$

The case for $S \equiv \Lambda X : K. T$ is similar to the last case. In all other cases, the proof follows by straightforward application of the induction hypothesis. \square

The operation *plus* does not have the usual properties under substitution; as following example shows, the equality

$$\text{plus}_{\Gamma_1, X \leq S : K_1, \Gamma_2}(T_2)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)] = \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(T_2[X \leftarrow T_1])$$

does not hold in general.

EXAMPLE 4.14 Consider the case where

$$\Gamma_1 \equiv Y \leq \top^* : \star, \quad \Gamma_2 \equiv \emptyset, \quad S \equiv Y, \quad T_1 \equiv Y, \quad \text{and} \quad T_2 \equiv X.$$

Then

$$\begin{aligned} \text{plus}_{Y \leq \top^* : \star, X \leq Y : \star}(X)[X \leftarrow \text{plus}_{Y \leq \top^* : \star}(Y)] &= (X + Y + \top^*)[X \leftarrow (Y + \top^*)] \\ &= Y + \top^* + Y + \top^*. \end{aligned}$$

On the other hand,

$$\text{plus}_{Y \leq \top^* : \star}(X[X \leftarrow Y]) = \text{plus}_{Y \leq \top^* : \star}(Y) = Y + \top^*.$$

We therefore need a lemma which says that the well-formed types are well-behaved under substitution with respect to the *plus* operation.

LEMMA 4.15 (*Substitution for plus*) If $\Gamma_1, X \leq S : K_1, \Gamma_2 \vdash T_2 : K_2$ and $\Gamma_1 \vdash T_1 : K_1$, then

$$\text{plus}_{\Gamma_1, X \leq S : K_1, \Gamma_2}(T_2)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)] \rightarrow_{\beta \wedge +} \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(T_2[X \leftarrow T_1]).$$

PROOF: By induction on the size of the derivation of $\Gamma_1, X \leq S : K_1, \Gamma_2 \vdash T_2 : K_2$. We proceed by case analysis on the form of T_2 .

$T_2 \equiv Y$. By the free variables lemma (lemma 3.17), $Y \in \text{dom}(\Gamma_1, X \leq S : K_1, \Gamma_2)$. Then there are three cases to consider.

$Y \in \text{dom}(\Gamma_1)$. Let $\Gamma_1 \equiv \Delta_1, Y \leq U : K, \Delta_2$. Then

$$\begin{aligned} & \text{plus}_{\Gamma_1, X \leq S : K_1, \Gamma_2}(Y)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)], \\ & \quad \text{by the definitions of } \textit{plus} \text{ and substitution,} \\ & = Y + (\text{plus}_{\Delta_1}(U)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)]) \\ & \quad \text{since } X \notin \text{FTV}(U) \cup \text{FTV}(\Delta_1), X \notin \text{FTV}(\text{plus}_{\Delta_1}(U)). \\ & = Y + \text{plus}_{\Delta_1}(U), \\ & = \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(Y[X \leftarrow T_1]). \end{aligned}$$

$Y \equiv X$. Then

$$\begin{aligned} & \text{plus}_{\Gamma_1, X \leq S : K_1, \Gamma_2}(X)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)], \\ & \quad \text{by the definitions of } \textit{plus} \text{ and substitution,} \\ & = \text{plus}_{\Gamma_1}(T_1) + (\text{plus}_{\Gamma_1}(U)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)]), \\ & \rightarrow_+ \text{plus}_{\Gamma_1}(T_1). \end{aligned}$$

On the other hand,

$$\begin{aligned} & \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(X[X \leftarrow T_1]) \\ & = \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(T_1), \\ & \quad \text{since } \text{FTV}(T_1) \cup \text{dom}(\Gamma_1), \text{ by strengthening for } \textit{plus}(4.12), \\ & = \text{plus}_{\Gamma_1}(T_1). \end{aligned}$$

$Y \in \text{dom}(\Gamma_2)$. Let $\Gamma_2 \equiv \Delta_1, Y \leq U : K, \Delta_2$. Then

$$\begin{aligned} & \text{plus}_{\Gamma_1, X \leq S : K_1, \Gamma_2}(Y)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)], \\ & \quad \text{by the definitions of } \textit{plus} \text{ and substitution,} \\ & = Y + (\text{plus}_{\Gamma_1, X \leq S : K_1, \Delta_1}(U)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)]), \\ & \quad \text{by generation (3.23) and the induction hypothesis,} \\ & \rightarrow_{\beta \wedge +} Y + \text{plus}_{\Gamma_1, \Delta_1[X \leftarrow T_1]}(U[X \leftarrow T_1]), \\ & = \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(Y[X \leftarrow T_1]). \end{aligned}$$

$T_2 \equiv \forall Y \leq S_1 : K.S_2$. Let $\Gamma \equiv \Gamma_1, X \leq S : K_1, \Gamma_2$. Then

$$\begin{aligned} & \text{plus}_{\Gamma}(\forall Y \leq S_1 : K.S_2)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)], \\ & \quad \text{by the definitions of } \textit{plus} \text{ and substitution,} \\ & = \forall Y \leq \text{plus}_{\Gamma}(S_1)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)] : K. \text{plus}_{\Gamma, Y \leq S_1 : K}(S_2)[X \leftarrow \text{plus}_{\Gamma_1}(T_1)], \\ & \quad \text{by generation (proposition 3.23) and the induction hypothesis,} \\ & \rightarrow_{\beta \wedge +} \forall Y \leq \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}(S_1[X \leftarrow T_1]) : K. \\ & \quad \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1], Y \leq S_1[X \leftarrow T_1] : K}(S_2[X \leftarrow T_1]) \\ & \quad \text{by the definitions of } \textit{plus} \text{ and substitution,} \\ & = \text{plus}_{\Gamma_1, \Gamma_2[X \leftarrow T_1]}((\forall Y \leq S_1 : K.S_2)[X \leftarrow T_1]). \end{aligned}$$

Other cases. All the other cases are similar to the case $T_2 \equiv \forall Y \leq S_1 : K.S_2$. \square

LEMMA 4.16 (*Monotonicity of plus with respect to $\rightarrow_{\beta\wedge}$*) If $\Gamma \vdash T : K$, then

1. $\Gamma \rightarrow_{\beta\wedge} \Gamma'$ implies $plus_{\Gamma}(T) \twoheadrightarrow_{\beta\wedge+} plus_{\Gamma'}(T)$.
2. $T \rightarrow_{\beta\wedge} T'$ implies $plus_{\Gamma}(T) \twoheadrightarrow_{\beta\wedge+}^{>0} plus_{\Gamma}(T')$.

PROOF: By simultaneous induction on the size of the derivation of $\Gamma \vdash T : K$. We proceed by case analysis on the form of T .

1. $\Gamma \rightarrow_{\beta\wedge} \Gamma'$.

$T \equiv X$. Let $\Gamma \equiv \Gamma_1, X \leq S : K_1, \Gamma_2$. Then we have to consider three cases.

- (a) $\Gamma_1 \rightarrow_{\beta\wedge} \Gamma'_1$. Then

$$plus_{\Gamma}(X) = X + plus_{\Gamma_1}(S)$$

by lemma 3.16 and part (1) of the induction hypothesis,

$$\twoheadrightarrow_{\beta\wedge} X + plus_{\Gamma'_1}(S) = plus_{\Gamma'_1}(X).$$

- (b) $S \rightarrow_{\beta\wedge} S'$. By lemma 3.16 and part (2) of the induction hypothesis.

- (c) $\Gamma_2 \rightarrow_{\beta\wedge} \Gamma'_2$. By the definition of *plus*.

$T \equiv \forall X \leq T_1 : K_1. T_2$. By generation for kinds (proposition 3.23), there are proper subderivations of $\Gamma \vdash T_1 : K_1$ and $\Gamma, X \leq T_1 : K_1 \vdash T_2 : \star$. Then, by part (1) of the induction hypothesis, it follows that

$$\begin{aligned} plus_{\Gamma}(T_1) &\twoheadrightarrow_{\beta\wedge} plus_{\Gamma'}(T_1), \text{ and} \\ plus_{\Gamma, X \leq T_1 : K_1}(T_2) &\twoheadrightarrow_{\beta\wedge} plus_{\Gamma', X \leq T_1 : K_1}(T_2). \end{aligned}$$

The result follows by the definitions of *plus* and $\twoheadrightarrow_{\beta\wedge}$.

Other cases. The rest of the cases are similar to the case $T \equiv \forall X \leq T_1 : K_1. T_2$, using generation for kinding (proposition 3.23) and part 1 of the induction hypothesis.

2. $T \rightarrow_{\beta\wedge} T'$.

$T \equiv \forall X \leq T_1 : K_1. T_2$. We have to consider three cases.

- (a) $T_1 \rightarrow_{\beta\wedge} T'_1$. By generation for kinding (proposition 3.23), there are proper subderivations of $\Gamma \vdash T_1 : K_1$ and $\Gamma, X \leq T_1 : K_1 \vdash T_2 : \star$. Then, by parts (2) and (1) of the induction hypothesis respectively, it follows that

$$\begin{aligned} plus_{\Gamma}(T_1) &\twoheadrightarrow_{\beta\wedge}^{>0} plus_{\Gamma}(T'_1), \text{ and} \\ plus_{\Gamma, X \leq T_1 : K_1}(T_2) &\twoheadrightarrow_{\beta\wedge} plus_{\Gamma, X \leq T'_1 : K_1}(T_2). \end{aligned}$$

The result follows by the definitions of *plus* and $\twoheadrightarrow_{\beta\wedge}$.

- (b) $T_2 \rightarrow_{\beta\wedge} T'_2$. By part (2) of the induction hypothesis.

- (c) $\forall X \leq T_1 : K_1. \bigwedge^* [S_1 \dots S_n] \rightarrow_{\beta\wedge} \bigwedge^* [\forall X \leq T_1 : K_1. S_1 \dots \forall X \leq T_1 : K_1. S_n]$.

$$\begin{aligned} &plus_{\Gamma}(\forall X \leq T_1 : K_1. \bigwedge^* [S_1 \dots S_n]) \\ &= \forall X \leq plus_{\Gamma}(T_1) : K. \bigwedge^* [plus_{\Gamma, X \leq T_1 : K_1}(S_1) \dots plus_{\Gamma, X \leq T_1 : K_1}(S_n)] \\ &\twoheadrightarrow_{\beta\wedge+} \bigwedge^* [\forall X \leq plus_{\Gamma}(T_1) : K. plus_{\Gamma, X \leq T_1 : K_1}(S_1) \dots \\ &\quad \dots \forall X \leq plus_{\Gamma}(T_1) : K. plus_{\Gamma, X \leq T_1 : K_1}(S_n)] \\ &= plus_{\Gamma}(\bigwedge^* [\forall X \leq T_1 : K_1. S_1 \dots \forall X \leq T_1 : K_1. S_n]) \end{aligned}$$

$T \equiv T_1 T_2$. We have to consider four cases.

- (a) $T_1 \rightarrow_{\beta\wedge} T'_1$,
- (b) $T_2 \rightarrow_{\beta\wedge} T'_2$,
- (c) $T \equiv \bigwedge^* [S_1 \dots S_n]$ and $\bigwedge^* [S_1 \dots S_n] T_2 \rightarrow_{\beta\wedge} \bigwedge^* [S_1 T_2 \dots S_n T_2]$.
- (d) $T \equiv \bigwedge X : K. S_1$ and $(\bigwedge X : K. S_1) T_2 \rightarrow_{\beta\wedge} S_1 [X \leftarrow T_2]$

Cases 2a, 2b, and 2c follow using similar arguments to those used for the case $T \equiv \forall X \leq T_1 : K_1.T_2$. Consider case 2d.

$$\begin{aligned}
& plus_{\Gamma}((\Lambda X : K.S_1) T_2) \\
&= (\Lambda X : K.plus_{\Gamma, X \leq \top K : K}(S_1)) plus_{\Gamma}(T_2) \\
&\rightarrow_{\beta \wedge} plus_{\Gamma, X \leq \top K : K}(S_1)[X \leftarrow plus_{\Gamma}(T_2)], \\
&\quad \text{by lemma 4.15,} \\
&\rightarrow_{\beta \wedge +} plus_{\Gamma}(S_1[X \leftarrow T_2]).
\end{aligned}$$

Other cases. The rest of the cases follows using a similar argument to the one used in the case $T \equiv \forall X \leq T_1 : K_1.T_2$. \square

LEMMA 4.17 Let $lub_{\Gamma}(S)$ be defined and $\Gamma \vdash S : K$. Then $plus_{\Gamma}(S) \rightarrow_{+}^{>0} plus_{\Gamma}(lub_{\Gamma}(S))$.

PROOF: By induction on the structure of S . Since $lub_{\Gamma}(S)$ is defined, it is enough to consider the following two cases.

$S \equiv X$. Let $\Gamma \equiv \Gamma_1, X \leq T : K, \Gamma_2$.

$$\begin{aligned}
plus_{\Gamma}(X) &= X + plus_{\Gamma_1}(T) \\
&= X + plus_{\Gamma}(T) && \text{by weakening (lemma 4.13),} \\
&\rightarrow_{+} plus_{\Gamma}(T) \\
&= plus_{\Gamma}(lub_{\Gamma}(X))
\end{aligned}$$

$S \equiv AT$. By the induction hypothesis. \square

Our measure to show the well-foundedness of $AlgF_{\lambda}^{\omega}$ considers the $\beta \wedge +$ -reduction paths of the *plus* versions of the types in the subtyping judgements. As we mentioned before, in ALGS-TVAR and ALGS-OAPP the types appearing in the hypothesis may be larger than those in their conclusions. Therefore, the well foundedness of the $AlgF_{\lambda}^{\omega}$ relation is not immediate. The next corollary gathers the previous results to serve our purposes.

COROLLARY 4.18

1. If $\Gamma \vdash X : K$, then $plus_{\Gamma}(X) \rightarrow_{\beta \wedge +}^{>0} plus_{\Gamma}(\Gamma(X))$.
2. If $\Gamma \vdash AT : K$ then $plus_{\Gamma}(AT) \rightarrow_{\beta \wedge +}^{>0} plus_{\Gamma}(lub_{\Gamma}(AT)^{nf})$.

PROOF: Item 1 is a particular case of the previous lemma (lemma 4.17), and item 2 is a consequence of lemma 4.17 and the monotonicity of *plus* with respect to $\rightarrow_{\beta \wedge +}$ (4.16(2)). \square

Finally, we can define our measure.

DEFINITION 4.19 (*Weight*)

1. $weight(\Gamma \vdash_{Alg} S \leq T) = \langle \max\text{-red}(plus_{\Gamma}(S)) + \max\text{-red}(plus_{\Gamma}(T)), \text{size}_j(\Gamma \vdash S \leq T) \rangle$,
2. $weight(\Gamma \vdash T : K) = \langle 0, 0 \rangle$,

where $\max\text{-red}(S)$ is the length of a maximal $\beta \wedge +$ -reduction path starting from S , and size_j is defined in definition 4.1.

Pairs are ordered lexicographically. Note that $\langle 0, 0 \rangle$ is the least *weight*.

PROPOSITION 4.20 (*Well-foundedness of $AlgF_{\lambda}^{\omega}$*)

If $\frac{J_1 \dots J_n}{J}$ is an $AlgF_{\lambda}^{\omega}$ rule, then $weight(J_i) < weight(J)$, for each $i \in \{1..n\}$.

PROOF: By inspection of the rules of $AlgF_{\lambda}^{\omega}$. \square

Finally, we can state the main result of this section.

THEOREM 4.21 (*Decidability of subtyping in F_{λ}^{ω}*)

For any context Γ and for any two types S and T , it is decidable whether $\Gamma \vdash S \leq T$.

4.3 Our decidability proof and full F_{\leq}

In the introduction we mentioned that subtyping in F_{\leq} , a second-order λ -calculus with bounded quantification defined by Curien and Ghelli in 1989, is undecidable. A question that comes to mind is: if we try to apply our proof of the decidability of subtyping in F_{λ}^{ω} to F_{\leq} , where will it fail?

If we consider the algorithm for the subtyping relation in [37], the place where our proof does not go through is when we try to prove that the algorithm terminates by calculating the maximal length of the *plus* versions of the types in the rule for subtyping quantified types. Remember that the subtyping rule for quantified types in full F_{\leq} is:

$$\frac{\Gamma \vdash T_1 \leq S_1 \quad \Gamma, X \leq T_1 \vdash S_2 \leq T_2}{\Gamma \vdash \forall X \leq S_1. S_2 \leq \forall X \leq T_1. T_2} \quad (F_{\leq}\text{-S-ALL})$$

Consider now the following case.

$$\begin{aligned} \Gamma &\equiv Y_4 \leq \top^*, Y_3 \leq Y_4, Y_2 \leq Y_3, Y_1 \leq Y_2, \\ T_1 &\equiv Y_1, \\ S_1 &\equiv \top^*, \\ T_2 &\equiv X \rightarrow X, \quad \text{and} \\ S_2 &\equiv X \rightarrow X. \end{aligned}$$

The *plus* versions of the types in the subtyping judgements of this example are as follows.

$$\begin{aligned} plus_{\Gamma, X \leq Y_1}(S_2) &\equiv (X + Y_1 + Y_2 + Y_3 + Y_4 + \top^*) \rightarrow (X + Y_1 + Y_2 + Y_3 + Y_4 + \top^*) \\ plus_{\Gamma, X \leq Y_1}(T_2) &\equiv (X + Y_1 + Y_2 + Y_3 + Y_4 + \top^*) \rightarrow (X + Y_1 + Y_2 + Y_3 + Y_4 + \top^*) \\ plus_{\Gamma}(\forall X \leq S_1. S_2) &\equiv \forall X \leq \top^*. (X + \top^*) \rightarrow (X + \top^*) \\ plus_{\Gamma}(\forall X \leq T_1. T_2) &\equiv \forall X \leq Y_1 + Y_2 + Y_3 + Y_4 + \top^*. \\ &\quad (X + Y_1 + Y_2 + Y_3 + Y_4 + \top^*) \rightarrow (X + Y_1 + Y_2 + Y_3 + Y_4 + \top^*) \end{aligned}$$

The length of a maximal $+$ -reduction in each case is:

$$\begin{aligned} \max\text{-red}(plus_{\Gamma, X \leq Y_1}(S_2)) &= 10 \\ \max\text{-red}(plus_{\Gamma, X \leq Y_1}(T_2)) &= 10 \\ \max\text{-red}(plus_{\Gamma}(\forall X \leq S_1. S_2)) &= 2 \\ \max\text{-red}(plus_{\Gamma}(\forall X \leq T_1. T_2)) &= 14. \end{aligned}$$

The *weight* of the conclusion $\Gamma \vdash \forall X \leq S_1. S_2 \leq \forall X \leq T_1. T_2$, as defined in definition 4.19, is smaller than the *weight* of the hypothesis $\Gamma, X \leq T_1 \vdash S_2 \leq T_2$, because the maximal length of a $+$ -reduction starting from the *plus* version of the conclusion is shorter than the maximal length of a $+$ -reduction starting from the *plus* version of that hypothesis. To be more precise,

$$\begin{aligned} \max\text{-red}(plus_{\Gamma}(\forall X \leq S_1. S_2)) + \max\text{-red}(plus_{\Gamma}(\forall X \leq T_1. T_2)) \\ < \\ \max\text{-red}(plus_{\Gamma, X \leq Y_1}(S_2)) + \max\text{-red}(plus_{\Gamma, X \leq Y_1}(T_2)). \end{aligned}$$

4.4 Decidability of type checking and type inference

We know from proposition 3.85 and theorem 3.92 that the algorithm *inf* is sound and computes minimal types for the F_{λ}^{ω} typing system.

The next step is to prove that the algorithm *inf* always terminates. This result completes the proof of decidability of type checking and type inference in F_{λ}^{ω} .

We first show that the auxiliary mapping *flub* is well-defined. To prove that *flub* is well-defined we use a similar argument to that used in section 4.2.1 to show that the relation defined by $AlgF_{\lambda}^{\omega}$ is well-founded. We show in lemma 4.24 that a maximal $\beta\lambda$ -reduction path of the *plus* version of the argument of *flub* is strictly longer than a maximal $\beta\lambda$ -reduction path of the *plus* version of the argument of its recursive call. But first we need to show the following auxiliary result.

LEMMA 4.22 If $\text{lub}_\Gamma^*(T)$ is defined, then $\Gamma \vdash T \leq \text{lub}_\Gamma^*(T)$.

LEMMA 4.23 Let $\text{lub}_\Gamma^*(T)$ be defined and $\Gamma \vdash T : K$. Then $\text{plus}_\Gamma(T) \rightarrow_{\beta\wedge+}^{>0} \text{plus}_\Gamma(\text{lub}_\Gamma^*(T))$.

PROOF: The proof follows by induction on the structure of T . If $T \equiv X$ or $T \equiv ST$, then the argument is the same as in lemma 4.17. The case remaining to be checked is when $T \equiv \bigwedge^K [T_1..T_n]$. Then

$$\begin{aligned} \text{plus}_\Gamma(\bigwedge^K [T_1..T_n]) &= \bigwedge^K [\text{plus}_\Gamma(T_1).. \text{plus}_\Gamma(T_n)] \\ \text{plus}_\Gamma(\text{lub}_\Gamma^*(\bigwedge^K [T_1..T_n])) &= \bigwedge^K [\text{plus}_\Gamma(T'_1).. \text{plus}_\Gamma(T'_n)], \end{aligned}$$

where $T'_1 \equiv T_i$ or $T'_i = \text{lub}_\Gamma^*(T_i)$. Since $\text{lub}_\Gamma^*(T)$ is defined, there exists $j \in \{1..n\}$ such that $\text{lub}_\Gamma^*(T_j)$ is defined. Now, for every k such that $\text{lub}_\Gamma^*(T_k)$ is defined, by the induction hypothesis, we have that

$$\text{plus}_\Gamma(T_k) \rightarrow_{\beta\wedge+}^{>0} \text{plus}_\Gamma(\text{lub}_\Gamma^*(T_k)).$$

Hence,

$$\text{plus}_\Gamma(\bigwedge^K [T_1..T_n]) \rightarrow_{\beta\wedge+}^{>0} \text{plus}_\Gamma(\text{lub}_\Gamma^*(\bigwedge^K [T_1..T_n])). \quad \square$$

LEMMA 4.24 (*Well-foundedness of flub*)

If $\Gamma \vdash T : K$, then $\text{flub}_\Gamma(T)$ is defined.

PROOF: If $\text{lub}_\Gamma^*(T^{nf})$ is undefined, flub terminates because $\rightarrow_{\beta\wedge}$ is strongly normalizing on well kinded types. Otherwise, define

$$\text{weight}(\text{flub}_\Gamma(T)) = \text{max-red}(\text{plus}_\Gamma(T)),$$

where $\text{max-red}(S)$ is the length of a maximal $\beta\wedge+$ -reduction path starting from S . Lemma 4.11 and the strong normalization property of $\rightarrow_{\beta\wedge+}$ imply that weight is well defined and always positive on well kinded types. Since $\text{lub}_\Gamma^*(T^{nf})$ is defined,

$$\begin{aligned} \text{plus}_\Gamma(T) &\rightarrow_{\beta\wedge+} \text{plus}_\Gamma(T^{nf}), && \text{by lemma 4.16(2),} \\ &\rightarrow_{\beta\wedge+}^{>0} \text{plus}_\Gamma(\text{lub}_\Gamma^*(T^{nf})), && \text{by lemma 4.23.} \end{aligned}$$

Then the weight of the arguments of flub reduces in each recursive call, which proves that flub is well-founded. \square

LEMMA 4.25 Let $\Gamma \vdash S, T : \star$ and $S =_{\beta\wedge} T$. Then $\text{flub}_\Gamma(S) \equiv \text{flub}_\Gamma(T)$.

We now define a measure for terms such that the type information inside the terms is considered to have constant value. The intuition behind the definition is to find a measure on terms which is invariant under type substitution (see lemma 4.27).

DEFINITION 4.26 (*size $\|-\|$*)

$$\begin{aligned} \|x\| &= 1, \\ \|\lambda x:T.e\| &= 1 + \|e\|, \\ \|e_1 e_2\| &= \|e_1\| + \|e_2\|, \\ \|\lambda X \leq T:K.e\| &= 1 + \|e\|, \\ \|e T\| &= 1 + \|e\|, \\ \|\text{for}(X \in T_1..T_n)e\| &= 1 + \|e\|. \end{aligned}$$

LEMMA 4.27 $\|e\| = \|e[X \leftarrow T]\|$.

PROPOSITION 4.28 (*Well-foundedness of inf*)

The inference rules for *inf* define a terminating algorithm.

PROOF: In the case of AT-VAR, the termination follows from the decidability of ok judgements (see corollary 4.3(1)). Furthermore, for each rule R of *inf*, if $\Gamma \vdash e : T$ is a hypothesis and $\Gamma \vdash e' : T'$ is the conclusion of R , then $\|e\| < \|e'\|$. Moreover, in the cases for AT-APP and AT-TAPP, $\Gamma \vdash f : T$ by the soundness of *inf* (proposition 3.85), $\Gamma \vdash T : \star$ by well-kindedness of typing (proposition 3.33). Hence $flub_{\Gamma}(T)$ is defined by lemma 4.24. Furthermore, *arrows* and *alls* define finite sets, and, as we proved in section 4.2.1, subtyping is decidable. Hence, the algorithm *inf* always terminates. \square

We can now state and prove that type checking in F_{\wedge}^{ω} is decidable.

THEOREM 4.29 (*Decidability of type checking in F_{\wedge}^{ω}*)

For any context Γ , and for any term e and type T closed in Γ , it is decidable whether $\Gamma \vdash e : T$.

PROOF: Infer a minimal type T' for e in Γ using *inf*, which is decidable by proposition 4.28, and check whether $\Gamma \vdash T' \leq T$, which is also decidable by theorem 4.21. \square

Every term e closed in a context Γ has type \top^* . We are interested in finding types other than \top^* , namely non-trivial types. Since *inf* computes minimal types and \top^* is the largest type (modulo $=_{\beta\wedge}$), if a term has a non trivial type in a given context, then the algorithm *inf* finds it.

THEOREM 4.30 (*Decidability of type inference in F_{\wedge}^{ω}*)

For any context Γ and for any term e closed in Γ , it is decidable whether there exists a type T such that $\Gamma \vdash e : T$ and $T \neq_{\beta\wedge} \top^*$.

PROOF: Infer a minimal type T for e in Γ using *inf*, which is decidable by proposition 4.28, and reduce T to normal form which is decidable because $\rightarrow_{\beta\wedge}$ is strongly normalising. Finally, check whether $T^{nf} \equiv \top^*$. \square

5 Conclusions

We defined the typed lambda calculus F_{\wedge}^{ω} , a natural generalization of Girard's system F^{ω} with intersection types and bounded polymorphism. A novel aspect of our presentation is the use of term rewriting techniques to present intersection types, which clearly splits the computational semantics (reduction rules) from the syntax (inference rules) of the system. We prove the Church-Rosser property of the reduction relation for types and terms in F_{\wedge}^{ω} and the strong normalization result for the reduction on types.

The normalized subtyping system, NF_{\wedge}^{ω} , is a significant technical contribution which is the key to finding a generation principle for the subtyping relation in F_{\wedge}^{ω} and an algorithm, $AlgF_{\wedge}^{\omega}$, which is shown to be equivalent to the original presentation. A generation principle is the key result needed to prove the Subject Reduction property for F_{\wedge}^{ω} , which we established in section 3.10.

We also presented a type inference algorithm and prove that it computes Minimal Types for F_{\wedge}^{ω} .

The main result of this paper is the decidability of F_{\wedge}^{ω} . A major component of the proof of decidability of typing in F_{\wedge}^{ω} is proving that the subtyping relation is decidable. A novel aspect of our proof is the use of a choice operator to model the behaviour of variables during subtype checking.

This paper contains the first proof of decidability of subtyping for a higher-order lambda calculus. The proof presented in [48] was developed afterwards and with knowledge of our proof. The decidability of subtyping is reduced to proving the strong normalization of the language of types enriched with a choice reduction. In section 4.3 we show where our proof of decidability breaks if applied to the undecidable second order system F_{\leq} . Because the decidability of subtyping is established for well-formed types, we also show that well-formation of types and contexts are decidable judgements. Finally, the proof of termination of typechecking uses the technology developed for the decidability of subtyping.

The work presented here has been extracted from the Ph.D thesis of the author [27], and a short version of the decidability of subtyping result been published in CSL'94 [26]. The techniques

developed here have been applied to study the combination of dependent types and subtyping in [3]. Martín Abadi and Luca Cardelli used these techniques in their book *A Theory of Objects* [1], demonstrating that our method extends naturally to a higher-order system with recursive types and object constructors. This work also influenced the work of Kathleen Fisher, who used this method in her dissertation (chapter 7 of [36]), and Gilles Barthe showed in [8] how decidability of typechecking in a calculus of order-sorted inductive types can be proved using our technique.

6 Acknowledgements

I want to express my gratitude to Mariangiola Dezani-Ciancaglini for her scientific and moral support which made the present work possible. I am also grateful to Henk Barendregt, Healfdene Goguen, James McKinna, Ugo de'Liguoro, and Benjamin Pierce for insightful discussions and useful comments on previews drafts. I also want to thank the anonymous reviewers for their useful comments and suggestions.

This research was supported by the Dutch organization for scientific research, NWO-SION project *Typed lambda calculus*, and by the European Community Training and Mobility of Researchers project *Theory And Applications Of Subtyping Systems For Proof Development, Programming, And Interactive Processes*.

References

- [1] M. Abadi and L. Cardelli. *A Theory of Objects*. Springer-Verlag, 1996.
- [2] R. M. Amadio and L. 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.
- [3] D. Aspinall and A. B. Compagnoni. Subtyping dependent types. In *Eleventh Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA., July 27-30 1996*. Preliminary version July 1995.
- [4] D. Aspinall and A. B. Compagnoni. Subtyping dependent types. *Theoretical Computer Science*, 266(1-2):273–309, September 2001.
- [5] H. Barendregt. Lambda calculi with types. In T. S. E. M. S. Abramsky, Dov M. Gabbay, editor, *Handbook of Logic in Computer Science*, volume 2, pages 117–309. Clarendon Press. Oxford, 1992.
- [6] H. P. Barendregt. *The Lambda Calculus*. North Holland, revised edition, 1984.
- [7] H. P. Barendregt, M. Coppo, and M. Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.
- [8] G. Barthe. Order-sorted inductive types. *Information and Computation*, 149(1):42–76, 25 February 1999.
- [9] G. Barthe and F. van Raamsdonk. Constructor subtyping in the calculus of inductive constructions. In J. Tiuryn, editor, *FOSSACS'00*, number 1784 in LNCS, pages 17–34, 2000.
- [10] V. Breazu-Tannen, T. Coquand, C. Gunter, and A. Scedrov. Inheritance as implicit coercion. *Information and Computation*, 93:172–221, 1991.
- [11] K. B. Bruce and J. Mitchell. PER models of subtyping, recursive types and higher-order polymorphism. In *Proceedings of the Nineteenth ACM Symposium on Principles of Programming Languages*, Albuquerque, NM, January 1992.
- [12] P. Canning, W. Cook, W. Hill, W. Olthoff, and J. Mitchell. F-bounded quantification for object-oriented programming. In *Fourth International Conference on Functional Programming Languages and Computer Architecture*, pages 273–280, September 1989.
- [13] L. Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988. Preliminary version in *Semantics of Data Types*, Kahn, MacQueen, and Plotkin, eds., Springer-Verlag LNCS 173, 1984.
- [14] L. Cardelli. Types for data-oriented languages. In *First Conference on Extending Database Technology*, volume 303 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1988.

- [15] L. Cardelli. Notes about $F_{<}$. Unpublished manuscript, October 1990.
- [16] L. Cardelli. Typeful programming. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts*. Springer-Verlag, 1991. An earlier version appeared as DEC Systems Research Center Research Report #45, February 1989.
- [17] L. Cardelli and J. Mitchell. Operations on records. *Mathematical Structures in Computer Science*, 1:3–48, 1991. Also in [42], and available as DEC Systems Research Center Research Report #48, August, 1989, and in the proceedings of MFPS '89, Springer LNCS volume 442.
- [18] L. Cardelli and P. Wegner. On understanding types, data abstraction, and polymorphism. *Computing Surveys*, 17(4), December 1985.
- [19] F. Cardone and M. Coppo. Two extensions of Curry's type inference system. In P. Odifreddi, editor, *Logic and Computer Science*, number 31 in APIC Studies in Data Processing, pages 19–76. Academic Press, 1990.
- [20] F. Cardone and M. Dezani-Ciancaglini. Combining type disciplines. *Annals of Pure and Applied Logic*, 66(3):197–230, 1994.
- [21] G. Castagna and B. Pierce. Decidable bounded quantification. In *Proceedings of Twenty-First Annual ACM Symposium on Principles of Programming Languages, Portland, OR*. ACM, January 1994.
- [22] G. Castagna and B. Pierce. Corrigendum: Decidable bounded quantification. In *Proceedings of the Twenty-Second ACM Symposium on Principles of Programming Languages (POPL), Portland, Oregon*. ACM, Jan. 1995.
- [23] G. Chen. Subtyping calculus of constructions. In *The 22nd International Symposium on Mathematical Foundations of Computer Science (MFCS)*, August 1997.
- [24] A. Church and J. B. Rosser. Some properties of conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936.
- [25] A. B. Compagnoni. Subtyping in F_{λ}^{ω} is decidable. Technical Report ECS-LFCS-94-281, LFCS, University of Edinburgh, January 1994.
- [26] A. B. Compagnoni. Decidability of higher-order subtyping with intersection types. In *Proceedings of the Annual Conference of the European Association for Computer Science Logic, CSL'94, Kazimierz, Poland*, number 933 in Lecture Notes in Computer Science. Springer-Verlag, June 1995. Preliminary version available as University of Edinburgh technical report ECS-LFCS-94-281, under the title “Subtyping in F_{λ}^{ω} is decidable”, January 1994.
- [27] A. B. Compagnoni. *Higher-Order Subtyping with Intersection Types*. PhD thesis, University of Nijmegen, The Netherlands, January 1995. ISBN 90-9007860-6.
- [28] A. B. Compagnoni. Subject reduction and minimal types for higher order subtyping. Technical Report ECS-LFCS-97-363, University of Edinburgh, LFCS, August 1997.
- [29] A. B. Compagnoni and H. H. Goguen. Typed operational semantics for higher order subtyping. *Information and Computation*, 184:242–297, August 2003.
- [30] A. B. Compagnoni and B. C. Pierce. Higher-order intersection types and multiple inheritance. *Mathematical Structures in Computer Science*, 6:469–501, 1996. Preliminary version available under the title *Multiple Inheritance via Intersection Types* as University of Edinburgh technical report ECS-LFCS-93-275 and Catholic University Nijmegen computer science technical report 93-18, Aug. 1993.
- [31] W. R. Cook, W. L. Hill, and P. S. Canning. Inheritance is not subtyping. In *Seventeenth Annual ACM Symposium on Principles of Programming Languages*, pages 125–135, San Francisco, CA, January 1990. Also in [42].
- [32] M. Coppo and M. Dezani-Ciancaglini. A new type-assignment for λ -terms. *Archiv. Math. Logik*, 19:139–156, 1978.
- [33] M. Coppo and M. Dezani-Ciancaglini. An extension of the basic functionality theory for the λ -calculus. *Notre-Dame Journal of Formal Logic*, 21(4):685–693, October 1980.
- [34] P.-L. Curien and G. Ghelli. Coherence of subsumption: Minimum typing and type-checking in $F_{<}$. *Mathematical Structures in Computer Science*, 2:55–91, 1992.
- [35] N. G. de Bruijn. Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. Math.*, 34(5):381–392, 1972.

- [36] K. Fisher. *Type Systems For Object-Oriented Languages*. PhD thesis, Computer Science, Stanford University, 1996. Supervisor: Prof. John Mitchell.
- [37] G. 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.
- [38] G. Ghelli and B. Pierce. Bounded existentials and minimal typing. *Theoretical Computer Science*, 193:75–96, 1998.
- [39] J.-Y. Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Université Paris VII, 1972.
- [40] H. Goguen. *A Typed Operational Semantics for Type Theory*. PhD thesis, University of Edinburgh, Aug. 1994.
- [41] H. Goguen. Soundness of a typed operational semantics for the logical framework. In *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, volume 1581 of *Lecture Notes on Computer Science*. Springer-Verlag, 1999.
- [42] C. A. Gunter and J. C. Mitchell. *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design*. The MIT Press, 1994.
- [43] J. R. Hindley. *The Church-Rosser property and a result in combinatory logic*. PhD thesis, University of Newcastle-upon-Tyne, 1964.
- [44] J. McKinna and R. Pollack. Pure type systems formalized. In M. Bezem and J. F. Groote, editors, *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, pages 289–305. Springer-Verlag, LNCS 664, Mar. 1993.
- [45] J. C. Mitchell. Toward a typed foundation for method specialization and inheritance. In *Proceedings of the 17th ACM Symposium on Principles of Programming Languages*, pages 109–124, January 1990. Also in [42].
- [46] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 2(42):223–243, 1942.
- [47] B. C. Pierce. *Programming with Intersection Types and Bounded Polymorphism*. PhD thesis, Carnegie Mellon University, December 1991. Available as School of Computer Science technical report CMU-CS-91-205.
- [48] B. C. Pierce and M. Steffen. Higher-order subtyping. *Theoretical Computer Science*, 176(1–2):235–282, 1997. Corrigendum in TCS vol. 184 (1997), p. 247.
- [49] B. C. Pierce and D. N. Turner. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming*, 4(2):207–247, Apr. 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".
- [50] J. Reynolds. Using category theory to design implicit conversions and generic operators. In N. D. Jones, editor, *Proceedings of the Aarhus Workshop on Semantics-Directed Compiler Generation*, number 94 in *Lecture Notes in Computer Science*. Springer-Verlag, January 1980. Also in [42].
- [51] J. C. Reynolds. Preliminary design of the programming language Forsythe. Technical Report CMU-CS-88-159, Carnegie Mellon University, June 1988.
- [52] W. W. Tait. Intensional interpretations of functionals of finite type i. *The Journal of Symbolic Logic*, 32(2):198–212, 1967.
- [53] J. Zwanenburg. *Object-Oriented Concepts and Proof Rules: Formalization in Type Theory and Implementation in Yarrow*. PhD thesis, Eindhoven University of Technology, 1999. Supervisor: Prof. Hemerik.