Typed Operational Semantics for Higher Order Subtyping

Adriana Compagnoni\textsuperscript{a} Healfdene Goguen\textsuperscript{b}

\textsuperscript{a}Stevens Institute of Technology, Castle Point on Hudson, Hoboken, NJ 07030, USA. abc@cs.stevens-tech.edu.
\textsuperscript{b}AT\&T Labs, 180 Park Ave., Florham Park, NJ 07932, USA. hhg@att.com.

Abstract

Bounded operator abstraction is a language construct relevant to object oriented programming languages and to ML2000, the successor to Standard ML. In this paper, we introduce $\mathcal{F}_\leq$, a variant of $\mathcal{F}_<$, with this feature and with Cardelli and Wegner’s kernel Fun rule for quantifiers. We define a typed operational semantics with subtyping and prove that it is equivalent with $\mathcal{F}_\leq$, using logical relations to prove soundness. The typed operational semantics provides a powerful and uniform technique to study metatheoretic properties of $\mathcal{F}_\leq$, such as Church–Rosser, subject reduction, the admissibility of structural rules, and the equivalence with the algorithmic presentation of the system that performs weak-head reductions.

Furthermore, we can show decidability of subtyping using the typed operational semantics and its equivalence with the usual presentation. Hence, this paper demonstrates for the first time that logical relations can be used to show decidability of subtyping.

\textit{Key words:} Subtyping; Type Theory; Typed Operational Semantics; Lambda Calculus; Dependent Kinds.

1 Introduction

During the last decade, object-oriented programming languages such as Smalltalk, C++, Modula 3, and Java have become popular because they encourage and facilitate software reuse and abstract design. In this time, the theoretical community has struggled to achieve a balance between safety and expressiveness of object-oriented programming languages, where safe languages use type systems to restrict the legal programs and thereby prevent errors, and expressive languages provide more constructs to allow the programmer to write programs more clearly or concisely.
A wide variety of language features has been proposed to model constructs from object-oriented programming languages in type systems, for example bounded quantification [29], recursive types [3], and matching [1, 11]. The feature we study in this paper is bounded abstraction on types, also called bounded operator abstraction. Cardelli and Harper are in favor of including this in ML2000 (private communication), the successor of Standard ML. The constructor is motivated by many examples due to Kim Bruce [8].

Consider the following example of extensible record types.

\[ \text{Extend } R \text{ with } [x: \text{Int}] \]

The intuition is that we want to extend the record type \( R \) with the field \( x: \text{Int} \). A necessary condition for such an extension is that \( R \) should be a record type. If \( R \) were \( \text{Bool} \), for example, the extended type would be nonsensical. Subtyping allows us to characterize all the types that are record types, because record types are subtypes of the empty record type. Furthermore, with type operator abstraction we can now define the function from record types to record types that, given a record type, creates a new record type extended with the field \( x: \text{Int} \).

\[ \vdash \Lambda R \leq [\cdot]: \ast. \text{Extend } R \text{ with } [x: \text{Int}] : \Pi R \leq [\cdot]: \ast. \ast \]

The semantics of \( \text{Extend} \) if \( x \) is already defined in \( R \) is beyond the scope of this paper, calculi to address this issue have been extensively studied [54, 18, 46, 16, 47].

The following example by Luca Cardelli shows how type operator abstraction allows us to express the type of sorting algorithms, ensuring that the elements being sorted belong to a domain having comparison operations \( eq \) and \( lt \).

\[
\text{Sortable } (A : \ast) = [eq : A \times A \to \text{Bool}, \lt : A \times A \to \text{Bool}]
\]

\[
\text{SortingAlgorithms } (S \leq \text{Sortable} : \ast \to \ast, A : \ast) = [\text{bubblesort}, \text{quicksort} : S(A) \times \text{Array}(A) \to \text{Array}(A)]
\]

\( \text{Sortable} \) is a type operator that given a type \( A \) creates a record type with methods \( \lt \) and \( eq \) on \( A \). Since \( S \) is a subtype of \( \text{Sortable} \), the record type \( S(A) \) has at least the methods \( \lt \) and \( eq \). Then a sorting algorithm such as \( \text{bubblesort} \), takes a record of methods containing methods \( \lt \) and \( eq \), an array of \( A \) and returns an array of \( A \). Hence, \( \text{SortingAlgorithms} \) is a bounded type operator that takes as argument a type operator less or equal \( \text{Sortable} \), and returns the record type with methods \( \text{quicksort} \) and \( \text{bubblesort} \).
This example shows how bounded operator abstraction allows us to express the type of sorting algorithms on types that may have more methods than just \( \text{lt} \) and \( \text{eq} \). In other words, we can type sorting algorithms on a larger collection of domain types.

Our framework for studying object-oriented programming is Abadi and Cardeli’s object calculi [2]. They have demonstrated that typed object calculi are well-suited to giving semantics for many features of object-oriented programming languages, such as class hierarchies, inheritance, self types, and binary methods. The language we study in this paper, \( \mathcal{F}_O \), is similar to their higher-order object calculus as presented in Chapter 20 of their book. We extend their language with bounded operator abstraction, a new feature that leads to metatheoretic difficulties, and we remove type formers, such as recursive and object types, that although important for the modeling of object-oriented programming languages do not significantly affect the metatheoretic development of the theory. The translation of object-oriented features might proceed as in Chapter 21 of their book if these features were reintroduced.

Our development is restricted to studying properties of the underlying object calculus, rather than particular object-oriented programming languages or features. Although a study of the semantic framework including properties such as subject reduction and an algorithm for type-checking is important to the understanding of objects, our work does not address such issues as type reconstruction that are clearly essential to a usable programming language. Indeed, it may be that an object-oriented programming language and its underlying object calculus have different metatheoretic properties, depending on the translation from the programming language to the calculus.

In the context of ML2000 [36], our work may be understood as giving a better understanding of possible mechanisms in a semantic framework for objects in ML, similar to work in type theory to explain module systems for ML [28, 38, 41, 42, 50]. There is some evidence to believe that type reconstruction and objects will interact well together [40], but clearly the integration of objects into ML is a large project that will require considerable research. Our study does introduce some ideas that could be relevant to studying object-oriented programming languages that include bounded operator abstraction, such as ML2000 or PolyTOIL [8].

The focus of this paper is on the metatheoretic treatment of subtyping. We see the contributions of the paper as the following:

- We give a logical relation style interpretation of subtyping, which allows us to study properties of kinding and subtyping simultaneously.
- We use this logical relation interpretation to show decidability of subtyping, the first use of logical relations for decidability of subtyping to occur in the
literature.
• We introduce a typed operational semantics for a language with subtyping, as an intermediate language for proving syntactic results about the type theory.
• We develop the metatheory of a particular type theory, $\mathcal{F}_{\leq}$, which captures important features for the foundations of object-oriented programming languages.

Furthermore, the typed operational semantics introduced in this paper was subsequently used to prove anti-symmetry of $\mathcal{F}_{\leq}$ [24]. This is the first proof of anti-symmetry of a system with higher-order subtyping, solving a long-standing open problem.

The paper is structured as follows. In the remainder of the introduction we give background information to clarify the above points. In Section 2 we introduce the syntax of $\mathcal{F}_{\leq}$. In Section 3 we introduce the typed operational semantics for this system. In Section 4 we develop the fundamental properties of types and kinds in $\mathcal{F}_{\leq}$. Section 5 gives the model construction that shows soundness of the typed operational semantics for the typing rules. Section 6 uses the previous results to prove subject reduction for terms in $\mathcal{F}_{\leq}$. Section 7 presents an algorithmic version of the system, where types are reduced only to weak-head normal form, and shows the equivalence of the usual and algorithmic presentations of $\mathcal{F}_{\leq}$. Section 8 shows decidability of the subtyping algorithm for $\mathcal{F}_{\leq}$. Finally, in Section 9 we summarize related and future work, and Section 10 gives our conclusions.

1.1 Metatheory of Subtyping and Logical Relations

We believe that type-checking for programming languages should be decidable. Decidable type systems prevent basic programming errors by limiting the meaningful programs. While we want a type system to be powerful to allow more expressive programs, it should also have a low overhead for the programmer. In particular, the compiler should be able to recognize correct and incorrect programs reliably without help from the programmer.

Decidability of type-checking for languages with subtyping relies on decidability of subtyping. Decidability of the subtyping algorithm is non-trivial, because the rule for subtyping variables recursively applies the subtyping algorithm to the bound of the variable. This means that subtyping consists of an interleaving of weak-head normalization and the replacement of variables by their bound, a process not bounded by the $\beta$-reduction of the $\lambda$-calculus.

Because our goal is to study the subtyping relation, we build our construction over the language of types and kinds in $\mathcal{F}_{\leq}$, rather than over terms and types.
Indeed, following our discussion above, while it should be possible to add non-terminating reductions to the language of terms, such as those for the object constructors of Abadi and Cardelli [2], it is our intention that the language of types and kinds should have desirable syntactic properties such as strong normalization. We can therefore use traditional approaches from type theory to study these properties.

In this paper we use logical relations for the first time to show decidability of subtyping. Logical relations have been used in traditional type theories without subtyping for a wide variety of applications, ranging from model theory to syntax. One of the most well-known applications is Tait’s proof of strong normalization for the $\lambda$-calculus with recursion on the natural numbers [52]. Because strong normalization is directly relevant to studies of termination and decidability of type-checking for type theory, it is not surprising that we can extend this technique to study decidability of subtyping.

The heart of our proof is the model construction outlined in Section 5. The basis of the proof is the same as Tait’s original logical-relation proof, but it also incorporates well-established ideas in dependent type theory, including partial interpretations [51], Kripke-style models for strong normalization [27], typed operational semantics [32, 33], and binary logical relations to interpret judgemental equality [26]. These extra techniques extend the logical relations proof so that it can incorporate subtyping and contexts, and thereby be used to show termination of subtyping.

The difficulty with decidability of subtyping, as mentioned above, is that the algorithm for subtyping is not bounded by the reductions of $\lambda$-calculus: there can be uses of the rule stating that a variable is less than its bound in a derivation of subtyping, which has no corresponding $\beta$-reduction. However, in the typing rules the bound has to be well-formed before the variable can be added to the context. Hence, if we are showing termination of the subtyping algorithm by induction on derivations of the typing judgement, analogous to the logical relations proof of strong normalization, we can always know the termination of the algorithm for the bound as a hypothesis to showing the termination of the algorithm for a variable. The logical-relation construction allows us to extend this argument from variables to arbitrary sequences of applications of variables.

In addition to following a traditional type-theoretic approach to showing decidability, our approach is conceptually much simpler than that used in existing proofs of the decidability of subtyping for systems of higher-order subtyping with bounded quantification. Other proofs have used reduction relations unrelated to the actual notion of computation of the type theory, for example the $+\text{-}reduction$ of Compagnoni [22, 23, 4, 5] or the $\Gamma\text{-}reduction$ of Pierce and Steffen [48]. Treating these auxiliary reduction relations leads to syntactic
complications unrelated to the basic problem of decidability.

Previous approaches to the metatheory of subtyping have also used strong
normalization of types as a basis for further reasoning about the subtyping
relation. For example, Compagnoni [23] defines a system for subtyping normal
types, and shows that this system is sound for rules of substitution and ap-
lication, relying on a previous result that every well-formed type is strongly
normalizing. In this paper, we instead build a logical-relation style interpreta-
tion of the subtyping relation together with the interpretation of the kinding
relation. We are then able to study metatheoretic properties of well-formed
types and the subtyping relation simultaneously. Because we are concentrating
on syntactic properties of subtyping, this does not follow the usual interpreta-
tion of subtyping in the literature as set inclusion [7, 9, 10, 12, 17, 25, 30, 49].

Our approach has the conceptual benefit of treating the interdependent judg-
ements of kinding and subtyping simultaneously, which means that it is not
sensitive to proving results in a specific order, and that it extends without
difficulty to bounded operator abstraction. Furthermore, the techniques we
use here were originally developed for dependent type theory. This suggests
that our proof technique will be well-suited to studying more complex type
theories with subtyping, such as the Calculus of Constructions with bounded
quantification.

Adding bounded operator abstraction to $F^\omega_{<\!}$ leads to complications in the
existing methods for developing the metatheory of subtyping. The new con-
structor means that subtyping is now needed to check well-formation of types.
Because type-checking is also needed in subtyping, this presents a circularity
that together with $\beta$-equality is not trivial to study. In particular, we now
need knowledge about subtyping to show results such as subject reduction for
types.

Most type systems with subtyping do not have this circularity: for example,$F^\omega_{<\!}$ [13, 14, 15, 17, 46], $F^\omega_{<\!}$ [25], and the systems in Abadi and Cardelli’s book
on objects [2] all separate the two judgements. Existing work on systems with
such a circularity [4] avoids the interdependency by finding a particular order
in which to prove results.

Some existing methods of studying the decidability of subtyping do not extend
easily to bounded operator abstraction. For example, the proof of termination
using $\Gamma$-reduction, which replaces a variable by its bound, does not extend triv-
ially to $F^\omega_{<\!}$, because the reduction is not confluent in the presence of bounded
operator abstraction. Pierce and Steffen’s proof relies on the confluence of $\beta$-
and $\Gamma$-reductions, which fails in the presence of bounded operators, as the
following example shows:

$$A \triangleleft_\beta (\Lambda X \leq A : K . A) B \triangleleft_\Gamma (\Lambda X \leq A : K . X) B \triangleright_\beta B$$
Alternatively, if $\Gamma$-reduction is not allowed under bounded abstraction then the $\xi$-rule for reduction under binders will no longer hold. Chen's proof of termination using $\Gamma$-reduction for the Calculus of Constructions [20] is not for bounded quantification, but the same problems would apply to any attempt to extend the technique.

Compagnoni's proof of termination using $+$-reduction probably extends to bounded operator abstraction. However, because $+$-reduction is not confluent, this method of showing termination is incompatible with using typed operational semantics for reasoning about the metatheory of a type theory. Using $+$-reduction here would mean the duplication of much of the model construction in order to show both soundness of the typed operational semantics and termination of $+$-reduction.

The decidability of type-checking follows straightforwardly from that of subtyping. Because the term structure of our language and $F^\omega_\leq$ [14, 17, 46] is the same (as opposed to the type structure), the proof is the same as for that system. We shall therefore only treat decidability of subtyping in this paper.

Another important property of a type system is subject reduction or type preservation, which states that evaluation of programs preserves their type. This is one of the central results of the paper. However, we also study on the same property at the level of types, as well as showing strong normalization for types, which states that type reduction will always terminate. Both of these properties are needed to show the correctness of the algorithms for type-formation and subtyping.

We now discuss the two steps of the typed operational semantics and the model construction in more detail, and mention which metatheoretic results follow from each step, with particular attention to the treatment of the subtyping judgement.

1.2 The Typed Operational Semantics

The intermediate system in our proof, the typed operational semantics, offers a powerful alternative induction principle to prove syntactic properties of type theories. Originally developed for type theories with dependent types, it gives a uniform method for showing the important metatheoretic properties of type theory, such as substitution and generation lemmas, strong normalization, subject reduction and Church-Rosser. By developing the metatheory of $F^\omega_\leq$, this paper demonstrates that the technique can be extended successfully to type theories with subtyping.

We give here a brief introduction to typed operational semantics, including
the aspects of particular interest in the development of the metatheory of subtyping. We refer the reader to the original papers on typed operational semantics [32, 33, 34] for a complete description of this technique.

Typed operational semantics define reduction to normal form for well-typed terms. The judgements are typically of the form \( \Gamma \vdash M \leadsto_w N \vdash_n P : A \), where the rules of inference enforce that \( N \) is the weak-head normal form of \( M \), that \( P \) is the normal form of \( M \), and that \( A \) is the type of \( M \). A typed operational semantics presents type theory from the perspective of computation instead of that of logical inference: we still need the full type information to derive the well-typedness of any term, but we replace the logical rules for application and abstraction by rules which instead express the reduction behavior of these terms in the calculus.

Properties such as Church–Rosser and subject reduction (Corollary 4.17) are particularly simple to show in the typed operational semantics. In the context of subtyping, we can also prove lemmas about replacing equal bounds and kinds in the context (Lemma 4.23), transitivity elimination (Lemma 4.30) and decidability in the typed operational semantics.

These properties are transferred to the logical presentation of the type theory by showing the equivalence of the logical presentation and the typed operational semantics. Soundness, that every term well-formed in the logical presentation is also well-formed in the typed operational semantics, follows the usual logical-relations style model for showing strong normalization, but where a type \( A \) is interpreted as a restricted set of terms of type \( A \) in the typed operational semantics. Completeness, that every term well-formed in the typed operational semantics is also well-formed in the logical presentation, is a direct induction on derivations of the typed operational semantics.

Another property that can be transferred from the typed operational semantics to the logical presentation is strong normalization. Strong normalization can be proved straightforwardly in the typed operational semantics by induction on derivations; by Soundness, the result also follows for the logical presentation.

The power of the technique is still more evident in systems with \( \eta \)-equality [32], because Church–Rosser is only true for the well-typed terms, and therefore cannot be shown by purely syntactic means [45]. We therefore intentionally avoid appealing to confluence on raw terms, because this property fails for such systems: our approach as it stands extends to them without fundamental difficulty. However, in this paper we choose not to study \( \eta \)-reduction, because it would distract from the principal ideas we wish to develop.

The presentation of subtyping in the typed operational semantics is motivated by existing algorithms for subtyping. An important aspect of such algorithms
is the ability to eliminate instances of transitivity in subtyping; transitivity leads to significant non-determinism, which in turn leads to infeasible subtyping algorithms. Thus existing algorithms for systems with decidable or semi-decidable subtyping [4, 22, 23, 49, 48] are syntax-directed in their search and only use transitivity in a specific, restricted way.

Our discussion above about decidability and the replacement of variables by their bound is reflected in the typed operational semantics by a particularly strong rule for type variables followed by a sequence of applications. One of the hypotheses of the rule of well-formation of $X(A_1, \ldots, A_n)$, in a context where $X$ is a type variable with bound $B$, is that $B(A_1, \ldots, A_n)$ must also be well-formed. This means that decidability of subtyping can be proved by induction on the derivations of well-formedness in the typed operational semantics: if we need to consider the case that a variable is less than its bound, we have the decidability of the bound as a hypothesis. This is the technique we use in Theorem 8.6.

The typed operational semantics is also syntax-directed, leading to strong inversion properties for the system. Because the typed operational semantics is an algorithmic presentation of the type theory, we are able to use the equivalence of the typed operational semantics with the usual typing rules to prove the generation lemmas in Section 5.4 that are the basis for the metatheory of the term language of $F_{\Sigma}$. This also allows us to prove in Section 7.6 the equivalence with the usual algorithmic presentation of the typechecking and subtyping relations, which include much less intermediate type information than typed operational semantics.

In our treatment, we have only given a typed operational semantics for the language of types and kinds, and the subtyping relation. This is because the full term language is intended to have recursion operators and objects, so the terms will not be strongly normalizing. The analysis of the language of types is still important, because it gives us information about the decomposition of subtyping judgements that allows us to prove subject reduction for terms and to show important properties about the typechecking and subtyping algorithms.

The strength of the logical-relations approach described in Section 1.1 is independent of the typed operational semantics. In particular, the same style of logical relation could be used for a proof of decidability of the algorithm for subtyping. Our proof uses typed operational semantics because the operational semantics led to the insight of using logical relations as the basis for showing decidability, and because it is the basis for a powerful and uniform technique for developing the full metatheory of type theory and not the single result of decidability.

9
1.3 The Model Construction

The logical relation construction that we use is somewhat more complicated than the usual models for strong normalization proofs. There are several reasons for this. First, the model captures both the typing and subtyping judgements. In contrast, most proofs of strong normalization only model the typing judgement, because equality can be understood by comparing the normal forms of the left- and right-hand sides. The model also needs to include context information in order to capture the replacement of a variable by its bound. Finally, in order to show soundness for the typed operational semantics the model needs to be formed from well-kindied objects, rather than being an untyped model as is often the case for strong normalization proofs.

We therefore build a model where a kind \( K \) is modeled as a family varying with respect to contexts \( \Delta \) of subsets of the types \( A \) such that \( A \) has kind \( K \) in \( \Delta \) in the typed operational semantics. We rely on techniques for including full type information that have been developed in the type-theory community:

- We introduce a partial interpretation of kinds [51]. The interpretation of \( \Pi X \leq A : K_1, K_2 \) is only defined if \( A \) is well-formed in the typed operational semantics of kind \( K_1 \). As part of the proof of soundness we show that if \( \Gamma \vdash A : K \) then the interpretation of \( K \) is defined, and so the interpretation is defined for all valid kinds of the language \( \mathcal{F}_\xi \). This partial interpretation is necessary because of the addition of bounded operator abstraction: the interdependency of types and kinds means that not all kinds are well-formed, which means that the interpretation needs to be undefined for those kinds that are not well-formed.
- We introduce a logical relation-style interpretation of the subtyping judgement as well as the kinding judgement, based on a similar treatment of judgemental equality by Coquand [26]. This allows us to lift the termination of the subtyping algorithm at types up to higher kinds in the same way that Tait’s logical relation for strong normalization lifts from the base type to higher types.
- We build a Kripke-style model [27] with contexts as possible worlds and context inclusion as the ordering. Whenever we need a fresh variable, for example in modeling \( \Lambda \)- or \( \Pi \)-binders, we can simply extend the associated context.

We shall discuss the technical aspects of these constructions when we define the model in Section 5.

Although this may seem to complicate the proof considerably, these techniques are all well-established in the dependent type theory community. Furthermore, these refinements of the definition of the logical relation are necessary
not only for soundness of the typed operational semantics but for the general logical relation argument for decidability of subtyping. The first point, the partial interpretation, is necessary for bounded operator abstraction. The logical relation construction for subtyping lifts termination of the algorithm up to higher kinds, and the Kripke-style model incorporates the information about contexts necessary for the replacement of variables by their bounds in the algorithm. It may have been the lack of general knowledge of these techniques that prevented such a proof from being discovered earlier.

We obtain an unexpected benefit by using the typed operational semantics and a model with kinded types: we are able to show the admissibility of the metatheoretic properties in Section 2.3, such as substitution, context replacement, and kind agreement, in the model construction, rather than showing them separately by induction on derivations. There is a simple intuition for why these structural rules can be interpreted when we extend the model to kinded types. First, we notice that every proof of strong normalization needs to allow for substitution properties, because it is exactly this that allows us to model $\beta$-reduction. Hence, it is not surprising that rules like substitution are sound for what is essentially a model of strongly normalizing types with kind information.

Although we say that the model is built with well-kindred types, the types are well-formed in the judgements of the typed operational semantics, a reduction sequence to normal form, not with respect to the judgements of $F^=_{\leq}$. Because the reduction includes kinding information, it is possible to prove completeness: that a derivation of the well-formedness of a type in the typed operational semantics gives rise to a derivation of well-formedness in the usual typing system.

We can show this completeness with respect to a restricted system $\vdash$ with no structural rules, such as substitution or thinning. Intuitively, this is because the rules of inference for the typed operational semantics are themselves restricted to rules for redexes such as $\beta$, and compatibility rules stating that reduction is a congruence with respect to the type formers. As usual for an algorithmic presentation, there are no rules of inference relating to substitution. This is in analogy with untyped reduction in $\lambda$-calculus, where we show that the substitution property, $M \rightarrow M'$ and $N \rightarrow N'$ implies $M[x\leftarrow N] \rightarrow M'[x\leftarrow N']$, is admissible, but it is not included as a rule of inference. Therefore, the rules of inference ST-BETA and the compatibility rules in the typed operational semantics have exact corresponding rules of inference for judgemental equality in $F^=_{\leq}$, unrelated to the structural rules.

Hence, by appealing to soundness (Corollary 5.13), which eliminates uses of structural rules in constructing a derivation in the typed operational semantics, and completeness (Proposition 4.12), which reflects the derivation without
uses of structural rules back into $\mathcal{F}_{\leq}$, we are able to eliminate all instances of these rules.

Goguen [35] abstracts the model construction demonstrated here to a simple Kripke-style model, where each type is interpreted in context $\Delta$ as the set of terms typeable in $\vdash$ in all extended contexts $\Delta, \Delta'$. Since the model used in this paper already includes this functionality, we have no need to perform the two model constructions separately. The more abstract study of the model construction led to improvements in the proof over earlier uses of typed operational semantics. For example, we believed that the Prefix Lemma for $\mathcal{F}_{\leq}$ (Lemma 6.2) was needed in the proof of Soundness (Corollary 5.13), but we were able to simplify the statement of Lemma 5.12 so as to remove it. This means that the Prefix Lemma can be established by equivalence and the same property for the typed operational semantics, instead of needing to do a separate induction.

An alternative approach would be to prove the equivalence of the systems with and without the structural rules directly. Doing this directly is conceptually simple but technically quite difficult, involving many structural lemmas such as the “splitting lemmas” [39], saying that if $\Gamma \vdash A =_\beta B : K$ then $\Gamma \vdash A : K$. Furthermore, because our system has many rules of inference, individual proofs of the structural lemmas will be long, tedious and error-prone. We instead show these properties by showing that they are valid in the model. Indeed, properties such as substitution must be valid for the model or it would be impossible to show strong normalization for $\beta$-reduction, which uses substitution fundamentally.

We have therefore reduced the metatheory of a type theory to essentially two steps: first, develop some basic results of the system in the typed operational semantics, where syntactic results are relatively easy; and secondly, prove the equivalence of the typed operational semantics with the typing rules, where completeness can be proved by a straightforward induction on derivations.

2 Syntax

We now present the rules for kinding, subtyping, and typing in $\mathcal{F}_{\leq}$. The rules are presented as simultaneously defined inductive relations with the following
judgement forms:

\[
\begin{align*}
\Gamma & \vdash \text{ok} & \text{well-formed context} \\
\Gamma & \vdash K & \text{well-formed kind} \\
\Gamma & \vdash K =_\beta K' & \text{kind equality} \\
\Gamma & \vdash A : K & \text{well-kinded type} \\
\Gamma & \vdash A =_\beta B : K & \text{type equality} \\
\Gamma & \vdash A \leq B : K & \text{subtype} \\
\Gamma & \vdash M : A & \text{well-typed term}.
\end{align*}
\]

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

2.1 Syntactic Categories

The \textit{kinds} of \( \mathcal{F}_\leq \) are the kind \(*\) of proper types and the kinds \( \Pi X \leq A : K_1.K_2 \) of functions on types (sometimes called type operators).

\[
K ::= * \quad \text{types} \\
\Pi X \leq A : K.K \quad \text{type operators}
\]

The language of \textit{types} of \( \mathcal{F}_\leq \) 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 \( X \); function types \( A \to B \); and polymorphic types \( \forall X \leq A : K.B \), in which the bound type variable \( X \) ranges over all subtypes of the upper bound \( A \). Moreover, like \( F^\omega \), we allow types to be abstracted on types, but we also allow bounds on the abstraction \( \Lambda X \leq A : K.B \). We can also apply types to argument types \( A B \); in effect, these forms introduce a simply typed \( \lambda \)-calculus with subtyping at the level of types. We shall sometimes use the word “types” to mean types and type operators.

The capture-avoiding substitution of \( A \) for \( X \) in \( B \) is written \( B[X \leftarrow A] \). We identify types that differ only in the names of bound variables. We shall write \( A(B_1, \ldots, B_n) \) for \(((A B_1) \ldots B_n)\). If \( A \) is of the form \( X(B_1, \ldots, B_n) \) then \( A \) has head variable \( X \). We write \( HV(\cdot) \) for the partial function returning the head variable of a type. We also extend the top type \( T_* \) to any kind \( K \) by defining inductively \( T_{\Pi X \leq A : K_1.K_2} = \Lambda X \leq A : K_1.T_{K_2} \).
$A := X$ type variable
$A \rightarrow A$ function type
$\forall X \leq A:K.A$ polymorphic type
$\Lambda X \leq A:K.A$ operator abstraction
$A A$ operator application
$T_*$ top type

The language of terms includes the variables ($x$), applications ($MN$), and functional abstractions ($\lambda x:A.M$) of the simply typed $\lambda$-calculus, as well as bounded type abstraction ($\lambda X \leq A:K.M$) and application ($MA$) of $F^\omega$. As in $F_\leq$, each type variable is given an upper bound at the point where it is introduced. We use the same notation for capture-avoiding substitution as that for types, and again identify $\alpha$-equivalent terms.

$M ::= x$ variable
$\lambda x:A.M$ abstraction
$M M$ application
$\lambda X \leq A:K.M$ type abstraction
$M A$ type application

The operational semantics of $F^\omega_\leq$ is given by the following reduction rules on terms and types.

**Definition 2.1 (Untyped Reduction)**

1. ($\lambda x:A.e_1)e_2 \rightarrow_{\beta_1} e_1[x \leftarrow e_2]
2. ($\lambda X \leq A:K_1.e)B \rightarrow_{\beta_1} e[X \leftarrow B]
3. ($\Lambda X \leq A:K.B)C \rightarrow_{\beta_2} B[X \leftarrow C]

Each relation ($\rightarrow_{\beta_1}$ and $\rightarrow_{\beta_2}$) is extended to a compatible relation with respect to term or type formation. The reduction $\rightarrow_\beta$ is defined by $\rightarrow_{\beta_1} \cup \rightarrow_{\beta_2}$. We write $\rightarrow_R$ for the transitive and reflexive closure of $\rightarrow_R$ and $\equiv_R$ for the least equivalence relation containing $\rightarrow_R$ and closed under $\alpha$-equivalence.

**2.2 Contexts**

A context $\Gamma$ 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 \( x:A \); type variable bindings have the form \( X \leq A : K \), where \( A \) is the upper bound of \( X \) and \( K \) is the kind of \( A \).

\[
\begin{align*}
\Gamma &::= \emptyset \quad \text{empty context} \\
\Gamma, \ x:A &\quad \text{term variable declaration} \\
\Gamma, \ X \leq A : K &\quad \text{type variable declaration}
\end{align*}
\]

We call the set of term and type variables defined in a context \( \Gamma \) the \textit{domain} of \( \Gamma \), 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, context, or statement. 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 \( \Gamma \), where \( X \) is implicitly asserted to be in \( \text{dom}(\Gamma) \).

We now give the rules of inference for the system \( \mathcal{F}_\leq \).

### 2.3 Structural Rules

This section presents general structural rules for \( \mathcal{F}_\leq \). In fact, each of the rules is admissible, which we shall show when we prove the equivalence of this system with the typed operational semantics. We shall write \( \Gamma \vdash J \) for judgements in the restricted system without these rules.

In the following \( J \) is not a typing statement (\( J \neq M : A \)).

\[
\begin{align*}
\Gamma_1, \Gamma_2 \vdash J &\quad \Gamma_1 \vdash A : * \quad \text{x \notin \text{dom}(\Gamma_1, \Gamma_2)} \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \text{(THIN)} \\
\Gamma_1, \Gamma_2 \vdash J &\quad \Gamma_1 \vdash A : K \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \text{X \notin \text{dom}(\Gamma_1, \Gamma_2)} \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \text{(TTHIN)} \\
\Gamma_1, \Gamma_2 \vdash J &\quad \Gamma_1 \vdash A \leq B : K \\
\Gamma_1, \Gamma_2[ X \leftarrow A ] \vdash J[ X \leftarrow A ] &\quad \text{(SUBST)} \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \Gamma_1 \vdash A =_\beta B : * \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \text{(CONTEXT-Eq)} \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \Gamma_1 \vdash A =_\beta B : K \\
\Gamma_1 \vdash K &\quad \text{(CONTEXT-T-Eq)} \\
\Gamma_1, \ x:A, \Gamma_2 \vdash J &\quad \Gamma_1 \vdash A =_\beta B : K \\
\Gamma_1 \vdash K &\quad \text{(KIND-AGREEMENT)}
\end{align*}
\]
2.4 Context Formation

The context formation rules are:

\[ \emptyset \vdash \text{ok} \] (C-EMPTY)

\[ \Gamma \vdash A : \star \quad x \notin \text{dom}(\Gamma) \]
\[ \frac{\Gamma, x : A \vdash \text{ok}}{\Gamma \vdash A : \star} \] (C-VAR)

\[ \Gamma \vdash A : K \quad X \notin \text{dom}(\Gamma) \]
\[ \frac{\Gamma, X \leq A : K \vdash \text{ok}}{\Gamma \vdash A : K \vdash \text{ok}} \] (C-TVAR)

2.5 Kind Formation

The well-formed kinds are those derived with the following rules.

\[ \Gamma \vdash \text{ok} \]
\[ \frac{\Gamma \vdash \star}{\Gamma^* \vdash \star} \] (K-*)

\[ \Gamma, X \leq A : K_1 \vdash K_2 \]
\[ \frac{\Gamma \vdash \Pi X \leq A : K_1, K_2}{\Gamma^* \vdash \Pi X \leq A : K_1, K_2} \] (K-II)

2.6 Kind Equality

The interconvertibility of kinds is the propagation of the interconvertibility of types within kinds.

\[ \Gamma \vdash K \]
\[ \frac{\Gamma \vdash K =_{\beta} K'}{\Gamma \vdash K' =_{\beta} K'} \] (K-EQ-REFL)

\[ \Gamma \vdash K =_{\beta} K' \]
\[ \frac{\Gamma \vdash K' =_{\beta} K''}{\Gamma \vdash K =_{\beta} K''} \] (K-EQ-TRANS)

\[ \Gamma \vdash \text{ok} \]
\[ \frac{\Gamma \vdash \star =_{\beta} \star}{\Gamma \vdash \star =_{\beta} \star} \] (K-EQ-*)

\[ \Gamma \vdash K_1 =_{\beta} K'_1 \]
\[ \Gamma \vdash A =_{\beta} A' : K_1 \]
\[ \Gamma, X \leq A : K_1 \vdash K_2 =_{\beta} K'_2 \]
\[ \frac{\Gamma \vdash \Pi X \leq A : K_1, K_2 =_{\beta} \Pi X \leq A' : K'_1, K'_2}{\Gamma \vdash \Pi X \leq A : K_1, K_2 =_{\beta} \Pi X \leq A' : K'_1, K'_2} \] (K-EQ-II)
2.7 Type Formation

For each type constructor, we give a rule specifying how it can be used to build well-formed type expressions. The new rules for type formation are the ones that deal with bounded type abstraction (T-TABS), type application (T-TAPP), and kind conversion (T-CONV).

\[
\Gamma \vdash \text{ok} \\
\Gamma \vdash \text{ok} \\
\Gamma, X \leq A:K, \Gamma_2 \vdash \text{ok} \\
\Gamma_1, X \leq A:K, \Gamma_2 \vdash X : K \\
\Gamma \vdash A_1 : * \quad \Gamma \vdash A_2 : * \\
\Gamma \vdash A_1 \rightarrow A_2 : * \\
\Gamma, X \leq A_1:K \vdash A_2 : * \\
\Gamma \vdash \forall X \leq A_1:K.A_2 : * \\
\Gamma, X \leq A_1:K_1 \vdash A_2 : K_2 \\
\Gamma \vdash \Lambda X \leq A_1:K_1.A_2 : \Pi X \leq A_1:K_1.K_2 \\
\Gamma \vdash A : \Pi X \leq B:K_1.K_2 \quad \Gamma \vdash C \leq B : K_1 \\
\Gamma \vdash AC : K_2[X \leftarrow C] \\
\Gamma \vdash A : K \quad \Gamma \vdash K =_\beta K' \\
\Gamma \vdash A : K' \\
\]  

(T-TOP)

(T-TVAR)

(T-ARROW)

(T-ALL)

(T-TABS)

(T-TAPP)

(T-CONV)

2.8 Type Equality

The judgemental type equality is generated by the typed beta-equality rule (T-Eq-Beta). It is a congruence with respect to type formation, and incorporates kind equivalence so that equal kinds contain the same equality relation on types.

\[
\Gamma, X \leq A_1:K_1 \vdash A_2 : K_2 \\
\Gamma \vdash C \leq A_1 : K_1 \\
\Gamma \vdash (\Lambda X \leq A_1:K_1.A_2)C =_\beta A_2[X \leftarrow C] : K_2[X \leftarrow C] \\
\Gamma \vdash A : K \\
\Gamma \vdash A =_\beta A : K \\
\Gamma \vdash A =_\beta B : K \\
\Gamma \vdash B =_\beta A : K \\
\Gamma \vdash A =_\beta B : K \quad \Gamma \vdash B =_\beta C : K \\
\Gamma \vdash A =_\beta C : K \\
\]  

(T-Eq-BETA)

(T-Eq-Refl)

(T-Eq-Sym)

(T-Eq-Trans)

17
\[
\begin{align*}
\Gamma &\vdash A_1 =_\beta B_1 : * & \Gamma &\vdash A_2 =_\beta B_2 : * \\
\Gamma &\vdash A_1 \rightarrow A_2 =_\beta B_1 \rightarrow B_2 : * \\
\Gamma, X \leq A_1 : K &\vdash A_2 =_\beta B_2 : * & \Gamma &\vdash K =_\beta K' \\
\Gamma &\vdash \forall X \leq A_1 : K. A_2 =_\beta \forall X \leq B_1 : K'. B_2 : * \\
\Gamma &\vdash A_1 =_\beta B_1 : K_1 \\
\Gamma, X \leq A_1 : K_1 &\vdash A_2 =_\beta B_2 : K_2 \\
\Gamma &\vdash K_1 =_\beta K'_1 \\
\Gamma &\vdash \forall X \leq A_1 : K_1. A_2 =_\beta \forall X \leq B_1 : K'_1. B_2 : \Pi X \leq A_1 : K_1. K_2 \\
\Gamma &\vdash A =_\beta B : \Pi X \leq E : K_1. K_2 \\
\Gamma &\vdash C =_\beta D : K_1 \\
\Gamma &\vdash C \leq E : K_1 \\
\Gamma &\vdash AC =_\beta BD : K_2[X \leftarrow C] \\
\Gamma &\vdash A =_\beta B : K \\
\Gamma &\vdash K =_\beta K' \\
\Gamma &\vdash A =_\beta B : K'
\end{align*}
\] (T-EQ-ARROW) (T-EQ-ALL) (T-EQ-TABS) (T-EQ-TAPP) (T-EQ-CONV)

2.9 Subtyping

The subtyping rules are those of \( F^\alpha_\omega \) [13, 14, 15, 17, 46], except for those dealing with bounded type abstraction and type application shown below and the rule for subtyping the quantifier. We chose Cardelli and Wegner’s kernel Fun rule for quantifiers with equal bounds [19]. The contravariant rule for quantifiers renders the system undecidable, and transitivity elimination in the presence of such a rule in the higher-order case remains an open problem.

\[
\begin{align*}
\Gamma &\vdash A : K \\
\Gamma &\vdash A \leq T_K : K \\
\Gamma &\vdash A =_\beta B : K \\
\Gamma &\vdash A \leq B : K \\
\Gamma &\vdash A \leq B : K \\
\Gamma &\vdash B \leq C : K \\
\Gamma &\vdash A \leq C : K \\
\Gamma, X \leq A : K, \Gamma_2 &\vdash \text{ok} \\
\Gamma &\vdash X \leq A : K, \Gamma_2 \vdash X \leq A : K \\
\Gamma &\vdash B_1 \leq A_1 : * \\
\Gamma &\vdash A_2 \leq B_2 : * \\
\Gamma &\vdash A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2 : * \\
\Gamma, X \leq C : K &\vdash A \leq B : * \\
\Gamma &\vdash \forall X \leq C : K. A \leq \forall X \leq C : K. B : * \\
\Gamma, X \leq C : K_1 &\vdash A \leq B : K_2 \\
\Gamma &\vdash \forall X \leq C : K_1. A \leq \forall X \leq C : K_1. B : \Pi X \leq C : K_1. K_2
\end{align*}
\] (S-TOP) (S-CONV) (S-TRANS) (S-TVAR) (S-ARROW) (S-ALL) (S-TABS)
\[
\Gamma \vdash A \leq B : \Pi X \leq D : K_1. K_2 \quad \Gamma \vdash C \leq D : K_1
\]
\[
\Gamma \vdash AC \leq BC : K_2[X \leftarrow C]
\] (S-TAPP)
\[
\Gamma \vdash A \leq B : K \quad \Gamma \vdash K =_\beta K'
\]
\[
\Gamma \vdash A \leq B : K'
\] (S-K-CONV)

2.10 Term Formation

The term formation rules are those of the second-order calculus of bounded quantification with the difference that we include kind annotations in terms, types, contexts, and subtyping judgements.

\[
\Gamma_1, x: A, \Gamma_2 \vdash \text{ok}
\]
\[
\Gamma_1, x: A, \Gamma_2 \vdash x : A
\] (T-VAR)
\[
\Gamma, x: A \vdash M : B
\]
\[
\Gamma \vdash \lambda x : A . M : A \rightarrow B
\] (T-ABS)
\[
\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A
\]
\[
\Gamma \vdash MN : B
\] (T-APP)
\[
\Gamma, X \leq A : K \vdash M : B
\]
\[
\Gamma \vdash \lambda X \leq A : K . M : \forall X \leq A : K . B
\] (T-TABS)
\[
\Gamma \vdash M : \forall X \leq A : K . B \quad \Gamma \vdash C \leq A : K
\]
\[
\Gamma \vdash MC : B[X \leftarrow C]
\] (T-TAPP)
\[
\Gamma \vdash M : A \quad \Gamma \vdash A \leq B : *
\]
\[
\Gamma \vdash M : B
\] (T-SUB)

3 The Typed Operational Semantics

We now introduce the typed operational semantics for \( \mathcal{F}_{\leq} \), which is organized in five judgement forms.

\[
\Gamma \vdash S \text{ ok} \quad \text{well-formed context}
\]
\[
\Gamma \vdash S K \rightarrow_n K' \quad \text{kind normalization}
\]
\[
\Gamma \vdash S A \rightarrow_w B \rightarrow_n C : K \quad \text{type reduction}
\]
\[
\Gamma \vdash S A \leq_w B : K \quad \text{weak-head subtyping}
\]
\[
\Gamma \vdash S A \leq B : K \quad \text{subtyping}
\]

The informal meaning of these judgements is as follows. In \( \Gamma \vdash S K \rightarrow_n K' \), \( K' \) is the normal form of \( K \). In \( \Gamma \vdash S A \rightarrow_w B \rightarrow_n C : K \), \( B \) is the weak-head
normal form of $A$ and $C$ its normal form. In $\Gamma \vdash_S A \leq_w B : K$, $A$ and $B$ are in weak-head normal form, and in $\Gamma \vdash_S A \leq B : K$, $A$ and $B$ are arbitrary types or type operators.

**Definition 3.1 (Weak-Head Normal)**

$\Gamma \vdash_S A_1 \rightarrow A_2$, $\forall X \leq A : K . B$, and $\Lambda X \leq A : K . B$ are weak-head normal. $X(A_1, \ldots, A_n)$ is weak-head normal if $A_1, \ldots, A_n$ are in normal form.

In order to prove the admissibility of transitivity in the semantics, we need to consider a stronger definition of weak-head normal form. We consider expressions of the form $X(A_1, \ldots, A_n)$ weak-head normal only if each $A_i$ is fully normalized. It may be possible to strengthen the model in Section 5 and use the standard definition of this notion instead.

We use the following notations:

- $\Gamma \vdash_S A : K$ is notation for $\Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K$, for some $B, C$.
- $\Gamma \vdash_S K$ is notation for $\Gamma \vdash_S K \rightarrow_n K'$, for some $K'$.
- $\Gamma \vdash_S A \rightarrow_n B : K$ is notation for $\Gamma \vdash_S A \rightarrow_w A \rightarrow_n B : K$.
- $\Gamma \vdash_S A \rightarrow_w B : K$ is notation for $\Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K$, for some $C$.
- $\Gamma \vdash_S A \rightarrow_n B : K$ means $\Gamma \vdash_S A \rightarrow_w C \rightarrow_n B : K$, for some $C$.
- $\Gamma \vdash_S A, B \rightarrow_n C : K$ means $\Gamma \vdash_S A \rightarrow_n C : K$ and $\Gamma \vdash_S B \rightarrow_n C : K$.
- $\Gamma \vdash_S K, K' \rightarrow_n K''$ means $\Gamma \vdash_S K \rightarrow_n K''$ and $\Gamma \vdash_S K' \rightarrow_n K''$.

The rules are presented as simultaneously defined inductive relations.

### 3.1 Context Formation

$\emptyset \vdash_S \text{ok}$ \hspace{1cm} (**SC-EMPTY**)

$\Gamma \vdash_S A : x \quad x \not\in \text{dom}(\Gamma)$

$\Gamma, x : A \vdash_S \text{ok}$ \hspace{1cm} (**SC-VAR**)

$\Gamma \vdash_S A : K' \quad \Gamma \vdash_S K \rightarrow_n K' \quad x \not\in \text{dom}(\Gamma)$

$\Gamma, X \leq A : K \vdash_S \text{ok}$ \hspace{1cm} (**SC-TVAR**)

### 3.2 Kind Normalization

$\Gamma \vdash_S \text{ok}$ \hspace{1cm} (**SK-*)

$\Gamma \vdash_S * \rightarrow_n *$ \hspace{1cm} (**SK-II**)

$\Gamma \vdash_S K_1 \rightarrow_n K'_1 \quad \Gamma \vdash_S A \rightarrow_n B : K'_1 \quad \Gamma, X \leq A : K_1 \vdash_S K_2 \rightarrow_n K'_2$

$\Gamma \vdash_S \Pi X \leq A : K_1, K_2 \rightarrow_n \Pi X \leq B : K'_1, K'_2$
The context formation and kind normalization rules follow from modifications to the context formation and kind equality rules of the system in Sections 2.4 and 2.6. For example, in the type variable rule SC-TVAR the kind of A and the kind in the declaration of X are \( \beta \)-equal but not necessarily identical.

3.3 Type Reduction

\[
\Gamma \vdash S \text{ ok} \\
\Gamma \vdash S A \rightarrow_w B \rightarrow_n C : K' \\
\Gamma \vdash S K \rightarrow_n K' \\
\Gamma \vdash S C : K' (X \leq A : K) \in \Gamma \\
\Gamma \vdash S X \rightarrow_w X \rightarrow_n X : K' \\
\Gamma \vdash S A \rightarrow_w w X(A_1, \ldots, A_m) : \Pi X \leq C : K_1.K_2 \\
\Gamma \vdash S B \rightarrow_w E \rightarrow_n F : K_1 \\
\Gamma \vdash S E \leq_w C : K_1 \\
\Gamma \vdash S K_2[X \leftarrow B] \rightarrow_n K \\
\Gamma \vdash S \Gamma(X) \rightarrow_n D : K' \\
\Gamma \vdash S D(A_1, \ldots, A_m, F) : K \\
\Gamma \vdash S A_1 \rightarrow_n B_1 : * \\
\Gamma \vdash S A_2 \rightarrow_n B_2 : * \\
\Gamma \vdash S (A_1 \rightarrow A_2) \rightarrow_w (A_1 \rightarrow A_2) \rightarrow_n (B_1 \rightarrow B_2) : * \\
\Gamma \vdash S \forall X \leq A : K.B \rightarrow_w \forall X \leq A : K.B \rightarrow_n \forall X \leq C : K'.D : * \\
\Gamma \vdash S K_1 \rightarrow_n K'_1 \\
\Gamma \vdash S A \rightarrow_n C : K'_1 \\
\Gamma, X \leq A : K_1 \vdash S B \rightarrow_n D : K_2 \\
\Gamma \vdash S \lambda X \leq A : K_1.B \rightarrow_w \lambda X \leq A : K_1.B \rightarrow_n \lambda X \leq C : K'_1.D : \Pi X \leq C : K'_1.K_2 \\
\Gamma \vdash S B \rightarrow_w \lambda X \leq A : K_1.D : \Pi X \leq A' : K'_1.K_2 \\
\Gamma \vdash S C \leq A : K'_1 \\
\Gamma \vdash S D[X \leftarrow C] \rightarrow_w E \rightarrow_n F : K \\
\Gamma \vdash S K_2[X \leftarrow C] \rightarrow_n K \\
\Gamma \vdash S B C \rightarrow_w E \rightarrow_n F : K \\
\Gamma \vdash S \Gamma(X) \rightarrow_n D : K' \\
\Gamma \vdash S D(A_1, \ldots, A_m, F) : K
\]

The rules for type reduction combine kinding information and computational behavior in the form of weak-head and \( \beta \)-normal forms. For example, the rule for arrow types says how to obtain the weak-head and \( \beta \)-normal forms of \( (A_1 \rightarrow A_2) \) in * from those for \( A_1 \) and \( A_2 \) in *.

The rule of inference for well-formedness of type variables applied to a sequence of types includes the usual information about well-formedness of the applicand and applicator. We also add information in the premises, not found elsewhere in the literature, stating that the bound of the variable has a normal form, and that replacing the variable by this normal form in the subject of the judgement is well-typed \( \Gamma \vdash S \Gamma(X) \rightarrow_n D : K' \) and \( \Gamma \vdash S D(A_1, \ldots, A_m, F) : K \).
These new premises strengthen the induction hypothesis when reasoning by induction on derivations of the typed operational semantics. The extra information this represents is enough to prove the decidability of subtyping for $\mathcal{F}_{\leq}$ directly by induction on derivations. The subtyping algorithm exactly needs to consider the replacement of a variable by its bound when determining whether such a type is a subtype of another type, so the derivations in the operational semantics give a measure that is a bound for the algorithm. We then need to show that the strong premises can be satisfied in the proof of soundness: this is a new application of the Tait-Girard method of logical relations, and it turns out to be possible because of the bound information for variables in the context. The beta rule, besides uncovering the outermost redex of the application $BC$ and contracting it, finds the weak-head normal form $E$ and the normal form $F$. The premise $\Gamma \vdash_s K_2[X_{\leftarrow} C] \rightarrow_w K$ ensures that $BC$ and $D[X_{\leftarrow} C]$ have $\beta$-equal kinds, and the subtyping premise $\Gamma \vdash_s C \leq A : K_1^l$ enforces the well-formation of $BC$.

The subtyping relation is defined using two judgements: one deals with types in weak-head normal form ($\Gamma \vdash_s A \leq_w B : K$) and the other with arbitrary types ($\Gamma \vdash_s A \leq B : K$).

### 3.4 Weak-Head Subtyping

\[
\frac{\Gamma \vdash S A \rightarrow_n B : * \quad \text{HV}(A) \text{ undefined}}{\Gamma \vdash S A \leq W \top : *} \quad \text{(SWS-TOP)}
\]

\[
\frac{\Gamma \vdash S X(A_1, \ldots, A_m) \rightarrow_n C : K \\
\Gamma \vdash S \Gamma(X) \rightarrow_n B : K' \quad \Gamma \vdash S B(A_1, \ldots, A_m) \rightarrow_w E : K \\
\Gamma \vdash S E \leq_w A : K \\
\Gamma \vdash S A \neq X(A_1, \ldots, A_m)}{\Gamma \vdash S X(A_1, \ldots, A_m) \leq_w A : K} \quad \text{(SWS-TAPP)}
\]

\[
\frac{\Gamma \vdash S B_1 \leq A_1 : * \\
\Gamma \vdash S A_2 \leq B_2 : *}{\Gamma \vdash S A_1 \rightarrow A_2 \leq_w B_1 \rightarrow B_2 : *} \quad \text{(SWS-ARRROW)}
\]

\[
\frac{\Gamma, X \leq A_1 : K \vdash S A_2 \leq B_2 : * \\
\Gamma \vdash S A_1, B_1 \rightarrow_n C : K'' \\
\Gamma \vdash S K, K' \rightarrow_n K''}{\Gamma \vdash S \forall X \leq A_1 : K, A_2 \leq_w \forall X \leq B_1 : K', B_2 : *} \quad \text{(SWS-ALL)}
\]

\[
\frac{\Gamma, X \leq A_1 : K_1 \vdash S A_2 \leq B_2 : K_2 \\
\Gamma \vdash S K_1, K_1' \rightarrow_n K''_1 \\
\Gamma \vdash S A_1, B_1 \rightarrow_n C : K''_1}{\Gamma \vdash S \forall X \leq A_1 : K_1, A_2 \leq_w \forall X \leq B_1 : K_1', B_2 : \Pi X \leq C : K''_1, K_2} \quad \text{(SWS-TABS)}
\]

22
The weak-head subtyping rules are motivated by the algorithmic rules in [23]. The rules SWS-ARROW, SWS-ALL, and SWS-TABS are structural. The rule for the maximal type $\top$, has a side condition to ensure that the algorithm is deterministic, and applications are only handled by SWS-TAPP or SWS-REFL.

A particular instance of SWS-TAPP is the rule for type variables. To check if $\Gamma \vdash_s X \leq_W A : K$, we have to check that the bound of $X$ in $\Gamma$ is a subtype of $A$ and that $\Gamma \vdash_s \Gamma(X) \leq A : K$. The premise $\Gamma \vdash_s E \leq_W A : K$ enforces that $A$ be in weak-head normal form. The side condition $A \neq X$ is to ensure determinism; if $A \equiv X$, the judgement instead follows by reflexivity.

3.5 Subtyping

$$\Gamma \vdash_s A \rightarrow^w C : K \quad \Gamma \vdash_s B \rightarrow^w D : K \quad \Gamma \vdash_s C \leq_W D : K \quad \frac{}{\Gamma \vdash_s A \leq B : K} \tag{SS-INC}$$

There is no rule for transitivity of subtyping in the semantic rules, but transitivity is a property of the "operational" subtyping (Lemma 4.30). Moreover, the rule SWS-TAPP includes a step of transitivity along the bound of a variable in the context. We interleave weak-head normalization steps in the subtyping algorithm via SS-INC. An alternative formulation would weak-head normalize the arguments of the hypothesis in the rules for weak-head subtyping.

4 Metatheory for $\mathcal{F}_{\leq}^\omega$

In this section we prove fundamental properties about the typed operational semantics for $\mathcal{F}_{\leq}^\omega$.

As we discussed in the introduction, the typed operational semantics plays a similar role to the algorithm in the usual development of the metatheory by providing inversion principles. However, it also allows us to show results such as subject reduction and strong normalization for types.

**Definition 4.1 (Closed)**

1. A term $M$ is closed with respect to a context $\Gamma$ if $\text{FV}(M) \cup \text{FTV}(M) \subseteq \text{dom}(\Gamma)$.
2. A type $A$ is closed with respect to a context $\Gamma$ if $\text{FTV}(A) \subseteq \text{dom}(\Gamma)$.
3. A kind $K$ is closed with respect to a context $\Gamma$ if $\text{FTV}(K) \subseteq \text{dom}(\Gamma)$.
4. (Closed Context)
   - The empty context is closed.
• \( \Gamma \equiv \Gamma_1, X \leq A:K \) is closed if \( A \) and \( K \) are closed with respect to \( \Gamma_1 \) and \( \Gamma_1 \) is closed.
• \( \Gamma \equiv \Gamma_1, x:A \) is closed if \( A \) is closed with respect to \( \Gamma_1 \) and \( \Gamma_1 \) is closed.

Judgements are closed if each of the terms to the right of the turnstile is closed with respect to the context and the context is closed.

**Lemma 4.2 (Closure)** If \( \Gamma \vdash_S J \) then \( \Gamma \vdash_S J \) is closed.

**Proof:** By induction on derivations. \( \square \)

**Lemma 4.3** If \( \Gamma \vdash_S J \) then all variables in \( \text{dom}(\Gamma) \) are distinct.

**Proof:** By induction on derivations. \( \square \)

**Lemma 4.4 (Weak Head and Normal Forms)**

1. If \( \Gamma \vdash_S K \rightarrow_n K' \) then \( K' \) is in normal form.
2. If \( \Gamma \vdash_S A \rightarrow_w B \rightarrow_w C : K \) then \( B \) is in weak-head normal form and \( C \) and \( K \) are in normal form.

**Proof:** By simultaneous induction on derivations. \( \square \)

In general, we shall use complete induction on derivations, rather than only considering the induction hypothesis on direct subderivations. The following lemma is useful to find subderivations of valid contexts.

**Definition 4.5 (Prefix)** A context \( \Gamma_0 \) is a prefix of \( \Gamma_0, \Gamma_1 \).

**Lemma 4.6 (Prefix)** If \( \Gamma \vdash_S J \) and \( \Gamma_0 \) is a prefix of \( \Gamma \) then there is a (not necessarily strict) subderivation of \( \Gamma_0 \vdash_S \) ok.

**Proof:** By induction on derivations. \( \square \)

The usual approach to proving Thinning for Pure Type Systems \cite{[6]} fails to take into account that although terms are in \( \alpha \)-equivalent classes, these classes do not extend to derivations. In other words, the fact that \( \Lambda X \leq A : K.B \) is \( \alpha \)-equivalent to \( \Lambda Y \leq A : K.B[X \leftarrow Y] \) for \( Y \not\in \text{FTV}(B) \) does not imply that a derivation of \( \Gamma, X \leq A : K \vdash_S B : K' \) is \( \alpha \)-equivalent to a derivation of \( \Gamma, Y \leq A : K \vdash_S B[X \leftarrow Y] : K' \), because that would mean considering \( \alpha \)-equivalence on free variables. Hence, although the term \( \Lambda X \leq A : K.B \) in the judgement \( \Gamma \vdash_S \Lambda X \leq A : K.B : K' \) may be changed to an \( \alpha \)-equivalent term \( \Lambda Y \leq A : K.B[X \leftarrow Y] \) for \( Y \not\in \text{FTV}(B) \), any derivation of this judgement must have a fixed parameter \( Z \) such that \( \Gamma, Z \leq A : K \vdash_S B[X \leftarrow Z] : K'' \) is a subderivation. If we try to use Thinning to extend this derivation to a derivation of the judgement \( \Gamma, Z \leq B : K'' \vdash_S \Lambda X \leq A : K.B : K' \) then the induction hypothesis will fail, because a context with two occurrences of \( Z \) is not a valid context.
The following alternative technique for proving Thinning was inspired by McKinna and Pollack’s development of the metatheory of Pure Type Systems [44]. We prove a lemma allowing a weak form of parallel substitution that only allows variables in a valid context to be substituted for variables. In the above example, we can choose to substitute a variable fresh in \( \Gamma, Z \leq B:K'' \) for the parameter corresponding to the bound variable \( X \), which when bound becomes \( \alpha \)-equivalent to the original term. Thinning is a simple corollary that follows by using the identity substitution.

**Definition 4.7 (Parallel Type Substitution)** A parallel type substitution \( \gamma \) for \( \Gamma \) is an assignment of types to type variables in \( \text{dom}(\Gamma) \). We write \( \epsilon \) for the empty assignment, \( \gamma[X:=A] \) to extend \( \gamma \) to assign \( A \) to \( X \), and \( \gamma(X) \) for the value of the assignment at a variable. We also write \( A[\gamma] \) for the replacement of variables in \( A \) with the values in \( \gamma \), and similarly for kinds and judgements; the value of this is undefined if there is a variable in \( A \) not in \( \text{dom}(\gamma) \). We say that \( \gamma \) is a type substitution for \( \Gamma \) in \( \Delta \) if \( \Delta \vdash_S \) ok and \( \Delta \vdash_S \gamma(X) \leq A[\gamma] : K' \), where \( \Delta \vdash_S K[\gamma] \rightarrow_n K' \), for each \( X \) with \( \Gamma = \Gamma_1, X \leq A : K, \Gamma_2 \).

**Definition 4.8** A renaming is a type substitution of variables for variables. \( \gamma \) is a renaming for \( \Gamma \) in \( \Delta \) if \( \Delta \vdash_S \) ok, and for each \( X \leq A : K \in \Gamma, \gamma(X) \leq A[\gamma] : K[\gamma] \in \Delta \).

Notice that we do not define substitutions for term variables. This is because these variables cannot occur in types or kinds, and hence do not significantly affect the judgements of the typed operational semantics. Also, notice that renamings are not necessarily injective.

**Lemma 4.9 (Renaming)** If \( \Gamma \vdash_S J \) and \( \gamma \) is a renaming for \( \Gamma \) in \( \Delta \) then \( \Delta \vdash_S J[\gamma] \).

**Proof:** By induction on derivations. We show the case in which the last applied rule of the derivation of \( \Gamma \vdash_S J \) is ST-TABS.

We are given that \( \Gamma \vdash_S \Lambda X \leq A : K_1. B \rightarrow_w \Lambda X \leq A : K_1. B \rightarrow_n \Lambda X \leq C : K'_1. D : \Pi X \leq C : K'_1. K_2 \) follows from \( \Gamma \vdash_S K_1 \rightarrow_n K'_1, \Gamma \vdash_S A \rightarrow_n C : K'_1, \) and \( \Gamma, X \leq A : K_1 \vdash_S B \rightarrow_n D : K_2 \).

By the induction hypothesis, \( \Delta \vdash_S K_1[\gamma] \rightarrow_n K'_1[\gamma] \) and \( \Delta \vdash_S A[\gamma] \rightarrow_n C[\gamma] : K'_1[\gamma] \).

To apply the induction hypothesis to the other premise we need to extend \( \Delta \) and \( \gamma \) to construct a renaming \( \gamma' \) for \( \Gamma, X \leq A : K_1 \). Notice that \( X \) may already appear in \( \Delta \), therefore we cannot extend it with a declaration for \( X \) because it could yield an illegal context with repeated declarations. Let \( Z \not\in \text{dom}(\Delta) \), and define \( \gamma' = \gamma[X:=Z] \).
We now show that \( \gamma' \) is a renaming for \( \Gamma, X \leq A : K_1 \) in \( \Delta, Z \leq A[\gamma] : K_1[\gamma] \). By SC-TVAR, \( \Delta, Z \leq A[\gamma] : K_1[\gamma] \vdash_S \) ok. The next step is to show that for every \( Y \leq E : K \in \Gamma, X \leq A : K_1 \) it follows that \( \gamma'(Y) \leq E[\gamma'] : K[\gamma'] \in \Delta, Z \leq A[\gamma] : K_1[\gamma] \). We have two cases to consider.

\[ Y \equiv X. \] By Closure (Lemma 4.2), \( FTV(A) \subseteq \text{dom}(\Gamma) \) and \( FTV(K_1) \subseteq \text{dom}(\Gamma) \), therefore \( A[\gamma] \equiv A[\gamma'] \) and \( K_1[\gamma] \equiv K_1[\gamma'] \).

\( Y \in \text{dom}(\Gamma) \). \( \gamma(Y) \equiv \gamma'(Y) \), by the definition of extension. Because \( \gamma \) is a renaming, we know that \( \gamma(Y) \leq E[\gamma'] : K[\gamma] \in \Delta \subseteq \Delta, Z \leq A[\gamma] : K_1[\gamma] \). Because \( \Gamma \) is closed, \( FTV(E) \subseteq \text{dom}(\Gamma) \) and \( FTV(K) \subseteq \text{dom}(\Gamma) \). Consequently, \( E[\gamma] \equiv E[\gamma'] \) and \( K[\gamma] \equiv K[\gamma'] \).

Hence \( \gamma' \) is a renaming for \( \Gamma, X \leq A : K_1 \) in \( \Delta, Z \leq A[\gamma] : K_1[\gamma] \).

Then, by the induction hypothesis, \( \Delta, Z \leq A[\gamma] : K_1[\gamma] \vdash_S B[\gamma'] \rightarrow_n D[\gamma'] : K_2[\gamma'] \).

By the definition of substitution, \( (\Lambda X \leq A : K_1.B)[\gamma] \equiv \Lambda Z \leq A[\gamma] : K_1[\gamma].B[\gamma[X := Z]] \), and similarly for the other binders in the conclusion. Finally, the result follows by ST-ABS.

We write \( \vdash_S \Delta \supseteq \Gamma \) if \( \vdash_S \) ok, \( x : A \in \Gamma \) implies \( x : A \in \Delta \), and \( X \leq A : K \in \Gamma \) implies \( X \leq A : K \in \Delta \). Thinning, which says that judgements are monotonic with respect to context extension, now follows as a corollary of Renaming taking \( \delta \) to be the identity substitution.

**COROLLARY 4.10 (Thinning)** If \( \Gamma \vdash_S J \) and \( \vdash_S \Delta \supseteq \Gamma \) then \( \Delta \vdash_S J \).

**LEMMA 4.11 (Determinacy)**

1. If \( \Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K \) and \( \Gamma \vdash_S A \rightarrow_w D \rightarrow_n E : K' \) then \( B \equiv D, C \equiv E \) and \( K \equiv K' \).
2. If \( \Gamma \vdash_S K \rightarrow_n K' \) and \( \Gamma \vdash_S K \rightarrow_n K'' \) then \( K' \equiv K'' \).

**PROOF:** By induction on derivations.

We show one case, SK-\( \Pi \). By inversion of \( \Gamma \vdash_S \Pi X \leq A : K_1.K_2 \rightarrow_n K'' \) we know that \( K'' \equiv \Pi Y \leq B'' : K''_1.K''_2 \), with \( \Gamma \vdash_S K_1 \rightarrow_n K''_1, \Gamma \vdash_S A \rightarrow_n B'' : K''_2 \) and \( Y \leq A : K_1 \vdash_S K_2[X \leftarrow Y] \rightarrow_n K''_2 \). By the induction hypothesis \( K''_1 \equiv K''_1 \) and \( B'' \equiv B'' \). By Lemma 4.6 \( \Gamma, X \leq A : K_1 \vdash_S \) ok, so \( \text{id}_\Gamma[Y := X] \) is a renaming for \( \Gamma, Y \leq A : K_1 \) in \( \Gamma, X \leq A : K_1 \). Hence, by Renaming \( \Gamma, X \leq A : K_1 \vdash_S K_2 \rightarrow_n K''_2[Y \leftarrow X] \), so by the induction hypothesis \( K''_1 \equiv K''_1[Y \leftarrow X] \), and so \( \Pi X \leq B : K''_1.K''_2 \equiv \Pi Y \leq B'' : K''_1.K''_2 \).

As we mentioned in the introduction, we want to prove completeness with respect to a system without the structural rules in Section 2.3. We shall write \( \Gamma \vdash \neg \neg J \) for judgements in the restricted system without these rules.
PROPOSITION 4.12 (Completeness)

(1) \( \Gamma \vdash_S \text{ok} \) implies \( \Gamma \vdash \text{ok} \).

(2) \( \Gamma \vdash_S K \rightarrow_n K' \) implies \( \Gamma \vdash K \) and \( \Gamma \vdash K =_\beta K' \).

(3) \( \Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K \) implies \( \Gamma \vdash A : K \), \( \Gamma \vdash A =_\beta B : K \), \( \Gamma \vdash A =_\beta C : K \), and \( \Gamma \vdash K =_\beta K' \). Furthermore, if \( B \equiv X(A_1, \ldots, A_m) \) and \( \Gamma \vdash_S \Gamma(X) \rightarrow_n D : K' \) then \( \Gamma \vdash D(A_1, \ldots, A_m) =_\beta D(A_1, \ldots, A_m) : K \) and \( \Gamma \vdash X(A_1, \ldots, A_m) \leq D(A_1, \ldots, A_m) : K \).

(4) \( \Gamma \vdash_S A \leq_W B : K \) implies \( \Gamma \vdash A \leq B : K \).

(5) \( \Gamma \vdash_S A \leq B : K \) implies \( \Gamma \vdash A, B : K \), \( \Gamma \vdash A \leq B : K \) and \( \Gamma \vdash K =_\beta K \).

PROOF: By simultaneous induction on derivations. We proceed by case analysis on the last rule of the derivation, presenting here a few representative cases. The hypotheses in each case are left implicit and follow exactly the notation of the rules in Section 3.

(1) SC-EMPTY Immediate by C-EMPTY.

SC-VAR By the induction hypothesis \( \Gamma \vdash A : \ast \), so the result follows by C-VAR.

SC-TVAR By the induction hypothesis \( \Gamma \vdash A : K' \), and by the induction hypothesis \( \Gamma \vdash K =_\beta K' \). By the symmetry of kind equality and T-CONV, \( \Gamma \vdash A : K \), and, by C-TVAR, the result follows.

(2) SK-\ast By the induction hypothesis \( 1 \Gamma \vdash \text{ok} \). By K-\ast, \( \Gamma \vdash \ast \), and, by K-EQ-\ast, \( \Gamma \vdash \ast =_\beta \ast \).

SK-\Pi By the induction hypothesis \( 2, \Gamma, X \leq A : K_1 \vdash K_2 \), by the induction hypothesis \( 3, \Gamma \vdash \Pi X \leq A : K_1, K_2 \). Then, by K-\Pi, \( \Gamma \vdash \Pi X \leq A : K_1, K_2 \). By the induction hypothesis \( 3, \Gamma \vdash A =_\beta B : K_1 \), and by the induction hypothesis \( 2, \Gamma, X \leq A : K_1 \vdash K_2 =_\beta K_2' \). By K-EQ-\Pi, \( \Gamma \vdash \Pi X \leq A : K_1, K_2 =_\beta \Pi X \leq B : K_1, K_2' \).

(3) ST-TVAR By Lemma 4.6 and the induction hypothesis \( 1 \Gamma \vdash \text{ok} \), and by the induction hypothesis \( 2 \Gamma \vdash K =_\beta K' \). Hence \( \Gamma \vdash X : K \) by T-TVAR, so \( \Gamma \vdash X : K' \) by T-CONV, and \( \Gamma \vdash X =_\beta X : K' \) by T-EQ-REFL. Furthermore, \( \Gamma \vdash K' =_\beta K' \) by K-EQ-SYM and K-EQ-TRANS.

Finally, suppose that \( \Gamma \vdash_S \Gamma(X) \rightarrow_n D : K' \), then we need to show that \( \Gamma \vdash D =_\beta D : K' \) and \( \Gamma \vdash X \leq D : K' \). By the induction hypothesis \( 3 \Gamma \vdash \Gamma(X) =_\beta D : K' \). Hence, by T-EQ-SYM and T-EQ-TRANS \( \Gamma \vdash D =_\beta D : K' \). Furthermore, \( \Gamma \vdash X \leq \Gamma(X) : K \) by S-TVAR, \( \Gamma \vdash X \leq \Gamma(X) : K' \) by S-K-CONV, and \( \Gamma \vdash X \leq D : K' \) by S-CONV and S-TRANS.

ST-TAPP By the induction hypothesis \( 3, \Gamma \vdash A : \Pi X \leq C : K_1, K_2, \Gamma \vdash B =_\beta E : K_1 \) and \( \Gamma \vdash B =_\beta F : K_1 \). By the induction hypothesis \( 4, \Gamma \vdash E \leq C : K_1 \). By S-CONV, \( \Gamma \vdash B \leq E : K_1 \), and, by
S-TRANS, $\Gamma \vdash B \leq C : K_1$. By T-APP, $\Gamma \vdash AB : K_2[X\leftarrow B]$. By the induction hypothesis 2, $\Gamma \vdash K_2[X\leftarrow B] = \beta K_1$, and, by T-CONV, $\Gamma \vdash AB : K$. We now have to prove that $\Gamma \vdash AB = \beta X(A_1, \ldots, A_m, F) : K$. By the induction hypothesis 3, $\Gamma \vdash A = \beta X(A_1, \ldots, A_m) : \Pi X \leq C : K_1, K_2$ and $\Gamma \vdash B = \beta F : K_1$. By T-EQ-APP, $\Gamma \vdash AB = \beta X(A_1, \ldots, A_m, F) : K_2[X\leftarrow B]$, and by T-EQ-CONV, $\Gamma \vdash AB = \beta X(A_1, \ldots, A_m, F) : K$. Furthermore, $\Gamma \vdash K = \beta K$ by K-EQ-SYM and K-EQ-TRANS.

Finally, we need to show that if $\Gamma \vdash G(X) \Rightarrow_n G : K'$ then $\Gamma \vdash G(A_1, \ldots, A_m, F) = \beta G(A_1, \ldots, A_m, F) : K'$ and $\Gamma \vdash X(A_1, \ldots, A_m, F) \leq G(A_1, \ldots, A_m, F) : K'$.

Hence, suppose $\Gamma \vdash G(X) \Rightarrow_n G : K'$. By the induction hypothesis 3 $\Gamma \vdash G(A_1, \ldots, A_m) = \beta G(A_1, \ldots, A_m) : \Pi X \leq C : K_1, K_2$ and $\Gamma \vdash X(A_1, \ldots, A_m) \leq G(A_1, \ldots, A_m) : \Pi X \leq C : K_1, K_2$. By T-EQ-APP and T-EQ-CONV $\Gamma \vdash G(A_1, \ldots, A_m, B) = \beta G(A_1, \ldots, A_m, F) : K$, so by T-EQ-SYM and T-EQ-TRANS $\Gamma \vdash G(A_1, \ldots, A_m, F) = \beta G(A_1, \ldots, A_m, F) : K$.

By T-EQ-SYM and T-EQ-TRANS $\Gamma \vdash X(A_1, \ldots, A_m) = \beta X(A_1, \ldots, A_m) : \Pi X \leq C : K_1, K_2$ and $\Gamma \vdash B = \beta B : K_1$, and by S-CONV $\Gamma \vdash B \leq B : K_1$. Hence $\Gamma \vdash X(A_1, \ldots, A_m, B) \leq G(A_1, \ldots, A_m, B) : K_2[X\leftarrow B]$ by S-APP, and $\Gamma \vdash X(A_1, \ldots, A_m, B) = \beta X(A_1, \ldots, A_m, F) : K_2[X\leftarrow B]$. By the induction hypothesis 3 we know that $\Gamma \vdash G(A_1, \ldots, A_m, B) = \beta G(A_1, \ldots, A_m, F) : K_2[X\leftarrow B]$ by T-EQ-TRANS, so $\Gamma \vdash X(A_1, \ldots, A_m, F) \leq G(A_1, \ldots, A_m, F) : K_2[X\leftarrow B]$ by several uses of S-TRANS. Finally, $\Gamma \vdash X(A_1, \ldots, A_m, F) \leq G(A_1, \ldots, A_m, F) : K$ by S-K-CONV.

(4) SWS-TAPP By the induction hypothesis 3 we know that $\Gamma \vdash B(A_1, \ldots, A_m) = \beta E : K$, so $\Gamma \vdash B(A_1, \ldots, A_m) \leq E : K$ by S-CONV, and by the induction hypothesis 4 $\Gamma \vdash E \leq A : K$, so $\Gamma \vdash B(A_1, \ldots, A_m) \leq A : K$ by S-TRANS. Also by the induction hypothesis 3 we know that $\Gamma \vdash X(A_1, \ldots, A_m) \leq A : K$. Hence $\Gamma \vdash X(A_1, \ldots, A_m, F) \leq A : K$ by S-TRANS.

SWS-ALL By the induction hypothesis 5, $\Gamma, X \leq A_1 : K \vdash A_2 \leq B_2 : \star$ and $\Gamma, X \leq A_1 : K \vdash B_2 : \star$. Then, by S-ALL, $\Gamma \vdash \forall X \leq A_1 : K, A_2 \leq \forall X \leq A_1 : K, B_2 : \star$. We now want to prove using T-EQ-ALL that $\Gamma \vdash \forall X \leq A_1 : K, B_2 \leq \forall X \leq B_1 : K', B_2 : \star$. The result then follows by S-CONV and S-TRANS. For that we need:

(a) $\Gamma, X \leq A_1 : K \vdash B_2 = \beta B_2 : \star$, which follows from T-EQ-REFL.

(b) $\Gamma \vdash K = \beta K'$, which follows from the induction hypothesis 2, K-EQ-SYM and K-EQ-TRANS.

(c) $\Gamma \vdash A_1 = \beta B_1 : K$. By the induction hypothesis 2 and
K-EQ-SYM, $\Gamma \vdash K'' =_\beta K$. By the induction hypothesis 3, \\
$\Gamma \vdash A_1 =_\beta B_1 : K''$, and by T-EQ-Conv, $\Gamma \vdash A_1 =_\beta B_1 : K$.

(5) **SS-INC** By the induction hypothesis 3, $\Gamma \vdash A, B, C, D : K$, $\Gamma \vdash A =_\beta C : K$, $\Gamma \vdash B =_\beta D : K$, and $\Gamma \vdash K =_\beta K$. By S-Conv, \\
$\Gamma \vdash A \leq C : K$ and $\Gamma \vdash D \leq B : K$, and, by the induction hypothesis 4, $\Gamma \vdash C \leq D : K$. Finally, by S-Trans, it follows that \\
$\Gamma \vdash A \leq B : K$. \hfill $\Box$

In Section 5 we shall see how Soundness (Corollary 5.13) can be used together \\
with this result to show the admissibility of the structural rules.

**Lemma 4.13 (Adequacy)**

- If $\Gamma \vdash S K \rightarrow_n K'$ then $K \rightarrow_\beta K'$.
- If $\Gamma \vdash S A \rightarrow_w B \rightarrow_n C : K$ then $A \rightarrow_\beta B \rightarrow_\beta C$.

We use parallel reduction [43, 44] as a tool for proving subject reduction for \\
the typed operational semantics.

**Definition 4.14 (Parallel Reduction)** Parallel reduction $\Rightarrow$ is the least \\
relation over types and kinds defined by the following rules of inference.

\[
\begin{align*}
X & \Rightarrow X \tag{P-VAR} \\
A \Rightarrow A' & \quad K \Rightarrow K' & B \Rightarrow B' \quad \Lambda X \leq A : K. B \Rightarrow \Lambda X \leq A' : K'. B' \tag{P-LAMBDA} \\
A \Rightarrow A' & \quad B \Rightarrow B' \quad AB \Rightarrow A' B' \tag{P-APP} \\
A \Rightarrow A' & \quad B \Rightarrow B' \quad (\Lambda X \leq C : K. A)(B) \Rightarrow A'[X \leftarrow B'] \tag{P-BETA}
\end{align*}
\]

plus similar rules, allowing reduction on each of the subterms, for the other \\
type and kind formers.

Parallel reduction extends in the obvious way to contexts.

Parallel reduction is useful because it has good inversion properties while being \\
closed under the following rule of substitution:

\[
A \Rightarrow A' & \quad B \Rightarrow B' \quad A[X \leftarrow B] \Rightarrow A'[X \leftarrow B'] \tag{P-SUBST}
\]

The following proof uses this and other simple properties about parallel reduction. See Takahashi’s excellent account of parallel reduction [53] for more \\
details.
Lemma 4.15 If $\Gamma \vdash_S A \rightarrow_w B : \Pi X \leq C : K_1.K_2$ and $B$ is not an abstraction then $B \equiv X(B_1, \ldots, B_n)$.

Lemma 4.16 (Parallel Subject Reduction for Types and Kinds)

- If $\Gamma \vdash_S \text{ok}$ and $\Gamma \Rightarrow \Gamma'$ then $\Gamma' \vdash_S \text{ok}$.
- If $\Gamma \vdash_S K \rightarrow_n K'$, $\Gamma \Rightarrow \Gamma'$ and $K \Rightarrow K''$ then $\Gamma' \vdash_S K'' \rightarrow_n K'$.
- If $\Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K$, $\Gamma \Rightarrow \Gamma'$ and $A \Rightarrow A'$, then there is a $B'$ such that $B \Rightarrow B'$ and $\Gamma' \vdash_S A' \rightarrow_w B' \rightarrow_n C : K$.
- If $\Gamma \vdash_S A \leq_w B : K$, $\Gamma \Rightarrow \Gamma'$, $A \Rightarrow A'$, and $B \Rightarrow B'$ then $\Gamma' \vdash_S A' \leq_w B' : K$.

Proof: By induction on derivations. We consider several cases:

ST-TVAR Suppose $\Gamma \Rightarrow \Gamma'$ and $X \Rightarrow B$. By inversion $B \equiv X$. Also, clearly if $(X \leq A : K) \in \Gamma$ then $\Gamma \Rightarrow \Gamma'$ implies $A \Rightarrow A'$, $K \Rightarrow K''$ and $(X \leq A' : K'') \in \Gamma'$.

- If $\Gamma \vdash_S A' \rightarrow_n C : K'$ and $\Gamma' \vdash_S K'' \rightarrow_n K'$, and since $C \Rightarrow C$ by the induction hypothesis $\Gamma' \vdash_S C : K'$, so $\Gamma' \vdash_S X \rightarrow_w X \rightarrow_n X : K'$.

ST-TAPP Suppose $\Gamma \Rightarrow \Gamma'$ and $AB \Rightarrow G$. Clearly $A$ is not an abstraction, so by inversion $G \equiv A'B'$ with $A \Rightarrow A'$ and $B \Rightarrow B'$. Hence, by the induction hypothesis $\Gamma' \vdash_S A' \rightarrow_n H : \Pi X \leq C : K_1.K_2$ and $X(A_1, \ldots, A_m) \Rightarrow H$, and by Lemma 4.4 and Lemma 4.15 $X(A_1, \ldots, A_m)$ is normal and so $X(A_1, \ldots, A_m) \equiv H$. By further use of the induction hypothesis there is an $E'$ such that $\Gamma' \vdash_S B' \rightarrow_w E' \rightarrow_n F : K_1$ and $E \Rightarrow E'$; $\Gamma' \vdash_S K_2[X \leftarrow B'] \rightarrow_n K$; and $\Gamma' \vdash_S E' \leq_w C : K_1$. Hence, $\Gamma' \vdash_S A'B' \rightarrow_w X(A_1, \ldots, A_m, F) \rightarrow_n X(A_1, \ldots, A_m, F) : K$ by ST-TAPP.

ST-BETA Suppose $\Gamma \Rightarrow \Gamma'$ and $BC \Rightarrow G$. By inversion of the reduction there are two possibilities:

- $B \Rightarrow B', C \Rightarrow C'$, and $G \equiv B'C'$. Then by the induction hypothesis $\Gamma' \vdash_S B' \rightarrow_w \Lambda X \leq A'' : K''_1.K_2 \rightarrow \Pi X \leq A' : K'_1.K_2$ with $\Lambda X \leq A : K_1.D \Rightarrow \Lambda X \leq A' : K'_1.D$, and also $\Gamma' \vdash_S C' \leq A' : K'_1$, $\Gamma' \vdash_S K_2[X \leftarrow C'] \rightarrow_n K$, and there is an $E'$ such that $\Gamma' \vdash_S D'[X \leftarrow C'] \rightarrow_w E' \rightarrow_n F : K$ and $E \Rightarrow E'$. Hence $\Gamma' \vdash_S B' \rightarrow_w E' \rightarrow_n F : K$ with $E \Rightarrow E'$.

SWS-REFL By Lemma 4.4 and the definition of weak-head normal, $X(A_1, \ldots, A_m)$ is normal. Hence, $X(A_1, \ldots, A_m) \Rightarrow C$ implies $C \equiv X(A_1, \ldots, A_m)$, and by the induction hypothesis $\Gamma \Rightarrow \Gamma'$ implies $\Gamma' \vdash_S X(A_1, \ldots, A_m) \rightarrow_n B : K$. 

□
It is easy to show that $\rightarrow_\beta$ is included in $\Rightarrow$, and that $\Rightarrow$ is included in $\rightarrow_\beta$, so we have the following corollary:

**Corollary 4.17 (Subject Reduction for Types and Kinds)**

- If $\Gamma \vdash_S \text{ok}$ and $\Gamma \rightarrow_\beta \Gamma'$ then $\Gamma' \vdash_S \text{ok}$.
- If $\Gamma \vdash_S K$ and $K \rightarrow_\beta K'$ then $\Gamma \vdash_S K'$.
- If $\Gamma \vdash_S A \rightarrow_w B \rightarrow_w C : K$ and $A \rightarrow_\beta A'$ then there is a $B'$ such that $B \rightarrow_\beta B'$ and $\Gamma \vdash_S A' \rightarrow_w B' \rightarrow_w C : K$.
- If $\Gamma \vdash_S A \leq B : K$, $A \rightarrow_\beta A'$ then $\Gamma \vdash_S A' \leq B : K$, and if $\Gamma \vdash_S A \leq B : K$, $B \rightarrow_\beta B'$ then $\Gamma \vdash_S A \leq B' : K$.

This corollary incorporates both Subject Reduction and also Church–Rosser, because the normal form is preserved by any one-step reduction.

**Corollary 4.18 (Church–Rosser)**

- If $\Gamma \vdash_S K \rightarrow_n K'$, $K \rightarrow_\beta K_1$, and $K \rightarrow_\beta K_2$ then there is a $K''$ such that $K_1 \rightarrow_\beta K''$ and $K_2 \rightarrow_\beta K''$.
- If $\Gamma \vdash_S A \rightarrow_n C : K$, $A \rightarrow_\beta A_1$ and $A \rightarrow_\beta A_2$ then there is a $B$ such that $A_1 \rightarrow_\beta B$ and $A_2 \rightarrow_\beta B$.

**Proof:** By Subject Reduction (Corollary 4.17) and Adequacy (Lemma 4.13).

We now prove Strong Normalization, using Subject Reduction to help.

**Definition 4.19 (Strong Normalization)** Strong normalization for types, written $\text{SN}(A)$, is the least predicate closed under the following rule of inference:

\[
\frac{\text{for all } B. (A \rightarrow_\beta B) \implies \text{SN}(B)}{\text{SN}(A)} \quad \text{(SN-1)}
\]

and similarly for kinds.

Strong normalization is easily seen to be closed under $\rightarrow_\beta$-reduction.

**Lemma 4.20 (Strong Normalization for Types and Kinds)**

1. If $\Gamma \vdash_S K \rightarrow_n K'$ then $K$ is strongly normalizing.
2. If $\Gamma \vdash_S A \rightarrow_w B \rightarrow_w C : K$ then $A$ is strongly normalizing.

**Proof:** By induction on derivations.

**SK-\star** By SN-1 we need to show that if $\star \rightarrow_\beta K'$ then SN($K'$), which follows because $\star \rightarrow_\beta K'$ is impossible.

**SK-II** By the induction hypothesis we know that SN($A$), SN($K_1$) and SN($K_2$).
By induction on the derivations of these premises we show that $SN(\Pi X \leq A: K_1.K_2)$.

By $SN\text{-}1$ we need to show that if $\Pi X \leq A: K_1.K_2 \rightarrow_{\beta_2} K'$ then $SN(K')$.

There are three possible reductions, corresponding to the subterms of $\Pi X \leq A: K_1.K_2$, and each of these cases follows by the appropriate induction hypothesis.

**ST-TOP** By $SN\text{-}1$ and the impossibility of $T_\ast \rightarrow_{\beta_2} B$.

**ST-TVAR** By $SN\text{-}1$ and the impossibility of $X \rightarrow_{\beta_2} B$.

**ST-TAPP** By the induction hypothesis we know that $SN(A)$ and $SN(B)$. By induction on the derivations of these we show that if $\Gamma \vdash_s A \rightarrow_w D : \Pi X \leq C: K_1.K_2$ and $D$ is not an abstraction then $SN(AB)$. By $SN\text{-}1$ we need to show that if $AB \rightarrow_{\beta_2} G$ then $SN(G)$. Again, if $D$ is not an abstraction then $A$ is not an abstraction, so if $AB \rightarrow_{\beta_2} G$ then we have two cases:

- $A \rightarrow_{\beta_2} A'$. Then by Parallel Subject Reduction (Lemma 4.16) there is a $D'$ such that $\Gamma \vdash_s A' \rightarrow_w D' : \Pi X \leq C: K_1.K_2$ and $D \Rightarrow D'$. By Lemma 4.4 $\Gamma \vdash_s A \rightarrow_w D : \Pi X \leq C: K_1.K_2$ implies $D$ is weak-head normal, and clearly if $D$ is weak-head normal and not an abstraction and $D \Rightarrow D'$ then $D'$ is not an abstraction. Hence, by the induction hypothesis we know $SN(A'B)$.

- $B \rightarrow_{\beta_2} B'$. This follows directly by the induction hypothesis.

Finally, $\Gamma \vdash_s A \rightarrow_w X(A_1, \ldots, A_m) : \Pi X \leq C: K_1.K_2$, and $X(A_1, \ldots, A_m)$ is not an abstraction.

**ST-ARROW** By the induction hypothesis we know $SN(A_1)$ and $SN(A_2)$. By induction on the derivations of these we show $SN(A_1 \rightarrow A_2)$. By $SN\text{-}1$ we need to show that if $(A_1 \rightarrow A_2) \rightarrow_{\beta_2} C$ then $SN(C)$. By inversion either $A_1 \rightarrow_{\beta_2} C_1$ or $A_2 \rightarrow_{\beta_2} C_2$, and each case follows by the induction hypothesis.

**ST-TALL** Similar to the case for SK-II.

**ST-TABS** Similar to the case for SK-II.

**ST-BETA** By the induction hypothesis we know that $SN(B)$ and $SN(D[X \leftarrow C])$, and furthermore by inversion of the premise $\Gamma \vdash_s C \leq A : K_1'$ there is a subderivation of $\Gamma \vdash_s C : K_1'$, so by the induction hypothesis $SN(C)$. By induction on the derivations of $SN(B)$ and $SN(C)$ we show that $\Gamma \vdash_s B \rightarrow_w \Lambda X \leq A: K_1.D : \Pi X \leq A': K_1'.K_2$ and $SN(D[X \leftarrow C])$ imply $SN(BC)$. By $SN\text{-}1$, we need $BC \rightarrow_{\beta_2} G$ implies $SN(G)$. By inversion of $BC \rightarrow_{\beta_2} G$ there are three cases:

- $B \rightarrow_{\beta_2} B'$. Then by Parallel Subject Reduction (Lemma 4.16) there is an $H$ such that $\Gamma \vdash_s B' \rightarrow_w H : \Pi X \leq A': K_1'.K_2$ and $\Lambda X \leq A: K_1.D \Rightarrow H$. By inversion $H \equiv \Lambda X \leq A'': K_1''.D'$ with $A \Rightarrow A''$, $K_1 \Rightarrow K_1''$ and $D \Rightarrow D'$. Because $SN$ is closed under reduction we know $SN(D'[X \leftarrow C])$. Hence $SN(B'C)$ by the induction hypothesis.

- $C \rightarrow_{\beta_2} C'$. Then since $SN$ is closed under reduction $SN(D[X \leftarrow C'])$, so by the induction hypothesis $SN(B'C')$.

- $B \equiv \Lambda X \leq A'': K_1''.H$ and $BC \rightarrow_{\beta_2} H[X \leftarrow C]$. By inversion of $\Gamma \vdash_s B \rightarrow_w \Lambda X \leq A: K_1.D : \Pi X \leq A': K_1'.K_2$ we know that $B \equiv \Lambda X \leq A: K_1.D$,
so in particular $H \equiv D$. Hence $\text{SN}(H[X \leftarrow C])$ follows by assumption.

**SS-INC** By the induction hypothesis for $A$ and $B$. □

**Definition 4.21** We define reduction of contexts, written $\vdash_S \Gamma \rightarrow_n \Phi$, as the least relation closed under the following rules of inference:

$$
\vdash_S \emptyset \rightarrow_n \emptyset \quad \text{(SCN-EMPTY)}
$$

$$
\vdash_S \Gamma \rightarrow_n \Phi \quad \vdash_S A \rightarrow_n C : \ast \quad x \notin \text{dom}(\Gamma)
$$

$$
\vdash_S \Gamma, x : A \rightarrow_n \Phi, x : C \quad \text{(SCN-VAR)}
$$

$$
\vdash_S \Gamma \rightarrow_n \Phi \quad \vdash_S K \rightarrow_n K' \\
\Gamma \vdash_S A \rightarrow_n C : K' \quad X \notin \text{dom}(\Gamma)
$$

$$
\vdash_S \Gamma, X \leq A : K \rightarrow_n \Phi, X \leq C : K' \quad \text{(SCN-TVAR)}
$$

We write $\vdash_S \Gamma ; \Delta \rightarrow_n \Phi$ if $\vdash_S \Gamma \rightarrow_n \Phi$ and $\vdash_S \Delta \rightarrow_n \Phi$.

**Lemma 4.22**

1. If $\vdash_S \Gamma \rightarrow_n \Phi$ and $\vdash_S A \rightarrow_n C : \ast$, then $\vdash_S A \rightarrow_n C : \ast$.
2. If $\vdash_S \Gamma \rightarrow_n \Phi$ then $\vdash_S \Gamma \rightarrow_n \Phi$.
3. If $\vdash_S \Gamma ; \Delta \rightarrow_n \Phi$ and $(X \leq A : K) \in \Gamma$ then there are $\Gamma_0$, $\Gamma_1$, $\Delta_0$, $\Delta_1$, $\Delta_0$, $\Delta_1$, $K''$, $K''$, $B$ and $C$ such that $\vdash_S \Gamma \equiv \Gamma_0, X \leq A : K, \Gamma_1$; $\Delta \equiv \Delta_0, X \leq B : K', \Delta_1$; $\vdash_S \Delta_0 \rightarrow_n \Delta_0$; $\vdash_S A \rightarrow_n C : K''$; and $\Delta_0 \vdash_S B \rightarrow_n C : K''$.

**Lemma 4.23** (Context Conversion) If $\vdash_S \Gamma ; \Delta \rightarrow_n \Phi$ and $\vdash_S J$ then $\vdash_S J$.

**Proof:** By induction on derivations. We consider two representative cases:

**ST-VAR** We know that $(X \leq A : K) \in \Gamma$, so by Lemma 4.22 Case 3 we know that there are $\Gamma_0$, $\Gamma_1$, $\Delta_0$, $\Delta_1$, $K''$, $K''$, $B$ and $C$ such that $\vdash_S \Gamma \equiv \Gamma_0, X \leq A : K, \Gamma_1$ and $\Delta \equiv \Delta_0, X \leq B : K', \Delta_1$, where $\Gamma_0 \vdash_S A \rightarrow_n C : K''$, $\Gamma_0 \vdash_S K \rightarrow_n K''$, $\Delta_0 \vdash_S B \rightarrow_n C : K''$, and $\Delta_0 \vdash_S K'' \rightarrow_n K''$. By thinning $\vdash_S K \rightarrow_n K''$, $\Delta \vdash_S K'' \rightarrow_n K''$, and $\Delta \vdash_S B \rightarrow_n C : K''$.

We have a premise that $\Gamma \vdash_S K \rightarrow_n K'$, so by Determinacy $K' \equiv K''$. Finally, $\Delta \vdash_S C : K'$, so by the induction hypothesis, so $\Delta \vdash_S X \rightarrow_w X \rightarrow_n X : K'$ by ST-VAR.

**ST-TABS** By the induction hypothesis $\Delta \vdash_S A \rightarrow_n C : K'$ and $\Delta \vdash_S K_1 \rightarrow_n K_1$. Hence $\vdash_S \Gamma, X \leq A : K_1 \rightarrow_n \Phi, X \leq A : K_1 \rightarrow_n \Phi, X \leq C : K_1'$, and so by the induction hypothesis $\Delta, X \leq A : K_1 \vdash_S B \rightarrow_n D : K_2$. The result follows by ST-TABS. □

**Lemma 4.24** (Subtyping Conversion)

- Suppose that $\Gamma \vdash_S A \leq_w B : K$. Then:
• If $\Gamma \vdash_S A, A' \xrightarrow{w} E : K$ then $\Gamma \vdash_S A' \leq_W B : K$.
• If $\Gamma \vdash_S B, B' \xrightarrow{w} E : K$ then $\Gamma \vdash_S A \leq_W B' : K$.

- Suppose that $\Gamma \vdash_S A \leq B : K$. Then:
  - If $\Gamma \vdash_S A, A' \xrightarrow{w} E : K$ then $\Gamma \vdash_S A' \leq B : K$.
  - If $\Gamma \vdash_S B, B' \xrightarrow{w} E : K$ then $\Gamma \vdash_S A \leq B' : K$.

**Proof:** By induction on derivations. We show two interesting cases:

**SW-S-ALL** We consider the case that $\Gamma \vdash_S \forall X \leq A_1 : K. A_2, A'_w \xrightarrow{w} E : \ast$, where the other case is similar but simpler. By inversion of the derivation for $\forall X \leq A_1 : K. A_2$ we know that $E \equiv \forall X \leq E_1 \leq K_m. E_2$ with $\Gamma \vdash_S A_1 \xrightarrow{w} E_1 : K_m$, $\Gamma \vdash_S K \xrightarrow{w} K_m$, and $\Gamma, X \leq A_1 : K \vdash_S A_2 \xrightarrow{w} E_2 : \ast$. By determinacy $K_m \equiv K_m$ and $C \equiv E_1$. By inversion of the derivation that $\Gamma \vdash_S A'_w \xrightarrow{w} \forall X \leq E_1 \leq K_m. E_2 : \ast$ we know that $A' \equiv \forall X \leq A'_1 \leq K_m. A'_2$, $\Gamma \vdash_S A'_1 \xrightarrow{w} E_1 : K_m$, $\Gamma \vdash_S K_m \xrightarrow{w} K_m$ and $\Gamma, X \leq A'_1 : K_m \vdash_S A'_2 \xrightarrow{w} E_2 : \ast$. By context conversion we know that $\Gamma, X \leq A_1 : K \vdash_S A'_2 \xrightarrow{w} E_2 : \ast$, so by the induction hypothesis $\Gamma, X \leq A_1 : K \vdash_S A'_2 \leq B_2 : \ast$. Finally, by context conversion again $\Gamma, X \leq A'_1 : K_m \vdash_S A'_2 \leq B_2 : \ast$, and the result follows by SW-S-ALL.

**SS-INC** We consider the case that $\Gamma \vdash_S A \rightarrow_w C'_w \xrightarrow{w} E : K$ and $\Gamma \vdash_S A' \rightarrow_w C''_w \xrightarrow{w} E : K$, where the other case is similar. We know $C \equiv C'$ by determinacy. Furthermore, by adequacy and subject reduction $\Gamma \vdash_S C, C''_w \xrightarrow{w} E : K$, so by the induction hypothesis $\Gamma \vdash_S C'' \leq_W D : K$. Hence $\Gamma \vdash_S A' \leq B : K$ by SS-INC.

**Lemma 4.25** If $\Gamma \vdash_S A \leq_W B : K$ then there are $C$ and $D$ such that $\Gamma \vdash_S A \rightarrow_w C : K$ and $\Gamma \vdash_S B \rightarrow_w D : K$.

**Proof:** By induction on $\Gamma \vdash_S A \leq_W B : K$, using inversion of the premises for SW-S-ARROW, SW-S-ALL and SW-S-TABS.

**Lemma 4.26 (Reflexivity)** If $\Gamma \vdash_S A \rightarrow B : K$ then $\Gamma \vdash_S A \leq A : K$.

**Proof:** We show the stronger property, that if $\Gamma \vdash_S A \rightarrow B \rightarrow_n C : K$ then $\Gamma \vdash_S B \leq_W B : K$ and $\Gamma \vdash_S A \leq A : K$, by induction on derivations.

We consider the one non-trivial case, ST-TAPP. We have $\Gamma \vdash_S X(A_1, \ldots, A_m, F) \rightarrow_n X(A_1, \ldots, A_m, F) : K$ as the conclusion, so $\Gamma \vdash_S X(A_1, \ldots, A_m, F) \leq_W X(A_1, \ldots, A_m, F) : K$ by SW-S-REFL. Finally, $\Gamma \vdash_S A B \leq_W A B : K$ by SS-INC.

**Lemma 4.27** If $\Gamma \vdash_S A \rightarrow B : K$ then $\Gamma \vdash_S A : \Pi X \leq C : K_1, K_2$, $\Gamma \vdash_S B \leq C : K_1$ and $\Gamma \vdash_S K_2[X \rightarrow B] \rightarrow_n K$.

**Proof:** By inversion on the derivation of $\Gamma \vdash_S A \rightarrow B : K$, using Lemma 4.25 for ST-TAPP, and using subject reduction, adequacy and a simple inversion
LEMMA 4.28 If \( \Gamma \vdash S \Gamma(X)(A_1, \ldots, A_m) : K \) then \( \Gamma \vdash S X(A_1, \ldots, A_m) \rightarrow_n X(A'_1, \ldots, A'_m) : K \) for some \( A'_1, \ldots, A'_m \).

PROOF: By induction on \( m \), using Thinning, Determinacy and Lemma 4.27.

LEMMA 4.29 If \( \Gamma \vdash S \Gamma(X)(A_1, \ldots, A_m) : K \) then \( \Gamma \vdash S X(A_1, \ldots, A_m) \leq \Gamma(X)(A_1, \ldots, A_m) : K \).

PROOF: By definition of \( \Gamma(X) \), together with Lemma 4.28 and Adequacy, Subject Reduction and Reflexivity for subtyping in the semantics.

LEMMA 4.30 (Transitivity) If \( \Gamma \vdash S A \leq B : K \) and \( \Gamma \vdash S B \leq C : K \) then \( \Gamma \vdash S A \leq C : K \).

PROOF: We show the stronger property that:

- if \( \Gamma \vdash S B \leq C : K \) then:
  1. if \( \Gamma \vdash S A \leq B : K \) then \( \Gamma \vdash S A \leq C : K \) for all \( A \), and
  2. if \( \Gamma \vdash S C \leq D : K \) then \( \Gamma \vdash S B \leq D : K \) for all \( D \).
- if \( \Gamma \vdash S B \leq W C : K \) then:
  1. if \( \Gamma \vdash S A \leq W B : K \) then \( \Gamma \vdash S A \leq W C : K \) for all \( A \), and
  2. if \( \Gamma \vdash S C \leq W D : K \) then \( \Gamma \vdash S B \leq W D : K \) for all \( D \).

by induction on derivations. We show several cases:

SWS-TOP We prove Case 1 by induction on derivations that \( \Gamma \vdash S C \leq W A : \ast \), using the induction hypothesis and SWS-TAPP for SWS-TAPP, contradiction for SWS-REFL and SWS-TABS, and SWS-TOP for the other rules.

Case 2 follows by inversion of derivations such that \( \Gamma \vdash S T_\ast \leq W D : \ast \).

SWS-ALL We have as premises that \( \Gamma \vdash S K, K' \rightarrow_n K'' \), that \( \Gamma \vdash S A_1, B_1 \rightarrow_n C : K'' \), and that \( \Gamma, X \leq A_1 : K \vdash S A_2 \leq B_2 : \ast \).

1. Suppose \( \Gamma \vdash S D \leq W \forall X \leq A_1 : K.A_2 : \ast \). By induction on this we show that \( \Gamma \vdash S D \leq W \forall X \leq B_1 : K'.B_2 : \ast \).

SWS-TAPP By the second induction hypothesis and SWS-TAPP.

SWS-ALL We have that \( D \equiv \forall X \leq D_1 : K''' \rightarrow_n D_2 \), that \( \Gamma \vdash S D_1, A_1 \rightarrow_n E : K''', \Gamma \vdash S K, K''' \rightarrow_n K'''' \), and that \( \Gamma, X \leq D_1 : K''' \vdash S D_2 \leq A_2 : \ast \). By Determinacy \( C \equiv E \) and \( K'' \equiv K'''' \), so \( \Gamma \vdash S D_1, B_1 \rightarrow_n C : K'' \) and \( \Gamma \vdash S K''', K' \rightarrow_n K'' \). Hence, by Context Conversion \( \Gamma, X \leq A_1 : K \vdash S D_2 \leq A_2 : \ast \), by the first induction hypothesis (1) \( \Gamma, X \leq A_1 : K \vdash S D_2 \leq B_2 : \ast \), and so by Context Conversion again \( \Gamma, X \leq D_1 : K''' \vdash S D_2 \leq B_2 : \ast \). Hence \( \Gamma \vdash S \forall X \leq D_1 : K'''.D_2 \leq W \forall X \leq B_1 : K'.B_2 : \ast \).

35
(2) Suppose \( \Gamma \vdash_s \forall X \leq B_1; K', B_2 \leq W \ D : * \). By inversion of this we show that \( \Gamma \vdash_s \forall X \leq A_1; K, A_2 \leq W \ D : * \).

SWS-Top Then \( \Gamma \vdash_s \forall X \leq A_1; K, A_2 \to_n E : * \) by Lemma 4.25, so 
\( \Gamma \vdash_s \forall X \leq A_1; K, A_2 \leq W \ T_* : * \) by SWS-Top.

SWS-All We know that \( \Gamma \equiv \forall X \leq D_1; K^{\text{th}}.D_2 \). By Determinacy
\( \Gamma \vdash_s A_1, D_1 \to_n C : K'' \) and \( \Gamma \vdash_s K, K^{\text{th}} \to_n K'' \), and by
Context Conversion \( \Gamma, X \leq A_1; K \vdash_s B_2 \leq D_2 : * \). Hence
by the induction hypothesis (2) \( \Gamma, X \leq A_1; K \vdash_s A_2 \leq D_2 : * \),
and \( \Gamma \vdash_s \forall X \leq A_1; K, A_2 \leq W \ \forall X \leq D_1; K^{\text{th}}.D_2 : * \) by SWS-
ALL.

SS-INC Both cases follow by inversion of the assumption, Determinacy, the
appropriate induction hypothesis for \( \leq_W \), and SS-INC. \( \square \)

5 Soundness

In this section we show the most important result for the metatheory of \( \mathcal{F}_{\leq} \):
that the typed operational semantics is sound for the typing rules in Section 2.
As we discussed in Section 1.3, this proof is essentially similar to traditional
proofs of strong normalization, although it includes several technical modifications
allowing us to prove soundness instead of normalization.

5.1 The Interpretation

We begin by defining the interpretation of kinds \( K \) with respect to a type sub-
stitution \( \gamma \) in a context \( \Delta \). There are two components to the interpretation:
the first component is a set of types well-formed in \( \Delta \) with particular pro-
erties, and models the judgement \( \Gamma \vdash A : K \); the second component is a relation
on types in the first component, and models the judgement \( \Gamma \vdash A \leq B : K \).

Partial interpretations are common in defining the semantics of dependent
type theories [32, 51]. In our proof, we need a partial interpretation to guaran-
tee that the bound \( A \) is well-formed for each \( \Pi \)-constructor \( \Pi X \leq A : K_1, K_2 \).
This is information that can only be known when the proof itself is carried
out, not when we define the interpretation. We prove that the interpretation
of a kind \( K \) is always defined if \( K \) is well-formed according to the typing rules
of \( \mathcal{F}_{\leq} \).

We need to include type information in our model, because we are proving
soundness with respect to a system with types. Unfortunately, the approach
used for simpler type systems, to assume an infinite collection of variables of
each type [31], does not easily transfer to our system. The problem is that the
kinds cannot be enumerated separately from the variables, because the kind
\(\Pi X \leq A : K_1, K_2\) may include occurrences of variables in the type \(A\). Hence,
we build a Kripke-style model following Coquand and Gallier [27], where the
possible worlds are valid contexts \(\Delta \vdash_S \Delta' \geq \Delta\).

The interpretation satisfies conditions similar to the usual saturated set conditions
and properties lifted from the typed operational semantics, such as transitiv-
itvity elimination (Lemma 5.6); properties about Kripke-style models such as
monotonicity (Lemma 5.7); and the substitution property (Lemma 5.8).

**Definition 5.1 (Semantic Object)** \(A\) is a semantic object for \(\Gamma\) and \(K\) if
\(\Gamma \vdash_S A \leq T_K : K\).

We shall use implicitly that if \(A\) is a semantic object for \(\Gamma\) and \(K\) then \(\Gamma \vdash_S
A : K\), which follows by definition of \(\Gamma \vdash_S A \leq T_K : K\) and inversion.

**Definition 5.2 (Interpretation of Kinds)** We give a partial interpretation of
kinds for both judgements, \([-]\), \((-,-)\) for typing and \([-] \leq (-,-)\) for subty-
ping:

- The interpretation \([\ast] \gamma \Delta\) is well-defined if \(\Delta \vdash_S \Delta\) ok. The two components are:
  - \([\ast] \gamma \Delta = \{ A \mid A\) is a semantic object for \(\Delta\) and \(\ast\} \).
  - \([\ast] \leq \gamma \Delta = \{(A,B) \mid A\) and \(B\) are semantic objects for \(\Delta\) and \(\ast\), and \(\Delta \vdash_S
  B \leq B : \ast\} \).

- The interpretation \([\Pi X \leq A : K_1, K_2] \gamma \Delta\) is well-defined if the following condi-
tions hold:
  - there is a \(K'\) such that \(\Delta \vdash_S (\Pi X \leq A : K_1, K_2)[\gamma] \rightarrow K'\),
  - \([K_1] \gamma \Delta'\) is defined for any \(\vdash_S \Delta' \geq \Delta\), and
  - if \(\vdash_S \Delta' \geq \Delta\) and \((C,A[\gamma]) \in [K_1] \leq \gamma \Delta'\), then \([K_2] \gamma [X := C] \Delta'\) is well-
defined.

Under these circumstances, the two components are:

- \([\Pi X \leq A : K_1, K_2] \gamma \Delta\) is the set of \(B\) such that:
  - \(B\) is a semantic object for \(\Delta\) and \(K'\),
  - if \(\vdash_S \Delta' \geq \Delta\) and \((C,A[\gamma]) \in [K_1] \leq \gamma \Delta'\), then \(B \in [K_2] \gamma [X := C] \Delta'\).

- \([\Pi X \leq A : K_1, K_2] \leq \gamma \Delta\) is the set of \((B,C)\) such that:
  - \(B\) and \(C\) are in \([\Pi X \leq A : K_1, K_2] \gamma \Delta\),
  - \(\Delta \vdash_S B \leq C : K'\),
  - if \(\vdash_S \Delta' \geq \Delta\) and \((D,A[\gamma]) \in [K_1] \leq \gamma \Delta'\) then \((B \cdot D, C \cdot D) \in [K_2] \leq \gamma [X := D] \Delta'\).

**Definition 5.3 (Interpretation of Contexts)** We define a partial inter-
pretation of contexts:

- \([\emptyset] \Delta = \{\epsilon\}\). This is defined if \(\Delta \vdash_S \Delta\) ok.
- \([\Gamma, x : A] \Delta = \llbracket \Gamma \rrbracket \Delta\). This is defined if \(x \notin \text{dom}(\Gamma)\), \(\Delta \vdash_S \Delta\) ok and, for any
\( \vdash S \Delta' \succeq \Delta \), \([\Gamma]\Delta'\) is defined and \(A[\gamma] \in \llbracket \cdot \rrbracket\) for every \(\gamma \in [\Gamma]\Delta'\).

- \([\Gamma, X \leq A:K]\Delta = \{\gamma[X:=B] \mid \gamma \in [\Gamma]\Delta \text{ and } \langle B, A[\gamma] \rangle \in \llbracket K \rrbracket \leq \gamma \Delta \}\). This is defined if \(X \not\in \text{dom}(\Gamma)\), \(\vdash S \Delta' \succeq \Delta\) and, for any \(\vdash S \Delta' \succeq \Delta\), \([\Gamma]\Delta'\) is defined and \(A[\gamma] \in \llbracket K \rrbracket\) for every \(\gamma \in [\Gamma]\Delta'\).

### 5.2 Properties of the Interpretation

We need to establish a variety of properties about the interpretation before carrying out the soundness proof.

**Definition 5.4** We write \(\Delta \vdash S \gamma, \gamma' \rightarrow_n \gamma''\) if for all \(X \in \text{dom}(\gamma)\) there is a \(K'\) such that \(\Delta \vdash S \gamma(X), \gamma'(X) \rightarrow_n \gamma''(X) : K'\).

We first give some simple properties about the interpretation:

**Lemma 5.5 (Basic Properties)**

1. If \([K]\gamma\Delta\) is defined then there is a \(K'\) such that \(\Delta \vdash S K[\gamma] \rightarrow_n K'\).
2. If \([\Gamma]\Delta\) is defined then \(\Delta \vdash S \gamma\Delta\).
3. If \([K]\gamma\Delta\) is defined and \(\gamma'(X) \equiv \gamma(X)\) for all \(X \in \text{dom}(\gamma)\) then \([K]\gamma\Delta = [K]\gamma'\Delta\).
4. If \([K]\gamma\Delta\) and \([K]\gamma'\Delta\) are defined and \(\Delta \vdash S \gamma, \gamma' \rightarrow_n \gamma''\) then \([K]\gamma\Delta = [K]\gamma'\Delta\).

**Proof:** Properties 1 and 2 follow by straightforward induction on \(K\) and \(\Gamma\).

Property 3 follows by induction on \(K\), using the fact that \(A[\gamma] \equiv A[\gamma']\) for types \(A\) if \(\gamma(X) \equiv \gamma'(X)\) for all \(X \in \text{FTV}(A)\), and similarly for kinds.

Property 4 follows by induction on \(K\). The \(\star\) case is trivial. For \(\Pi X \leq A:K_1.K_2\), we know that \(\Delta \vdash S \Pi X \leq A:K_1.K_2[\gamma] \rightarrow_n K'\) and \(\Delta \vdash S \Pi X \leq A:K_1.K_2[\gamma] \rightarrow_n K''\). By Adequacy and Subject Reduction \(\Delta \vdash S \Pi X \leq A:K_1.K_2[\gamma'] \rightarrow_n K'\) and \(\Delta \vdash S \Pi X \leq A:K_1.K_2[\gamma''] \rightarrow_n K''\), and by Determinacy \(K' \equiv K''\). Hence \(B\) is a semantic object for \(K'\) iff \(B\) is a semantic object for \(K''\), \([K_1]\gamma\Delta' = [K_1]\gamma'\Delta'\) by the induction hypothesis, and if \(C \in [K_1]\gamma\Delta'\) then \([K_2]\gamma[X:=C]\Delta' = [K_2]\gamma'[X:=C]\Delta'\) by the induction hypothesis for any \(\vdash S \Delta' \succeq \Delta\), from which it follows easily that \(\Pi X \leq A:K_1.K_2[\gamma] = [\Pi X \leq A:K_1.K_2]\gamma\Delta.\)

Next, we need some properties similar to the usual saturated set conditions. In the following we assume \(\Delta \vdash S K[\gamma] \rightarrow_n K'\):

**Lemma 5.6 (Saturated Sets)**

1. If \(A \in \llbracket K \rrbracket\) then \(A\) is a semantic object for \(\Delta\) and \(K'\).
2. If \((A, B) \in \llbracket K \rrbracket \leq \gamma \Delta\) then \(\Delta \vdash S A \leq B : K'\).
(3) If \((A, B) \in \llbracket K \rrbracket \leq \gamma \Delta\) then \(A \in \llbracket K \rrbracket \gamma \Delta\) and \(B \in \llbracket K \rrbracket \gamma \Delta\).

(4) If \(A\) and \(B\) are in \(\llbracket K \rrbracket \gamma \Delta\) and \(\Delta \vdash S\ A, B \rightarrow_n C : K'\) then \((A, B) \in \llbracket K \rrbracket \leq \gamma \Delta\).

(5) If \((A, B)\) and \((B, C)\) are in \(\llbracket K \rrbracket \leq \gamma \Delta\) then \((A, C) \in \llbracket K \rrbracket \leq \gamma \Delta\).

(6) If \(\Delta(X)(A_1, \ldots, A_m) \in \llbracket K \rrbracket \gamma \Delta\) then \(X(A_1, \ldots, A_m) \in \llbracket K \rrbracket \gamma \Delta\) and \((X(A_1, \ldots, A_m), \Delta(X(A_1, \ldots, A_m))) \in \llbracket K \rrbracket \leq \gamma \Delta\).

(7) If \(\Delta \vdash S\ A, B \rightarrow_w C \rightarrow_n D : K'\) and \(A \in \llbracket K \rrbracket \gamma \Delta\) then \(B \in \llbracket K \rrbracket \gamma \Delta\).

(8) If \(A \in \llbracket K \rrbracket \gamma \Delta\) then \((A, T_{K[\gamma]}) \in \llbracket K \rrbracket \leq \gamma \Delta\).

**Proof:** Properties 1, 2 and 3 follow by construction.

Property 4 follows by induction on \(K\), using Reflexivity and Conversion of subtyping in the semantics, plus Adequacy, Subject Reduction and Determinism for the II case.

Property 5 follows by induction on \(K\), using Transitivity of subtyping in the semantics.

For Property 6, \(X(A_1, \ldots, A_m)\) is a semantic object for \(\Delta\) and \(K'\), using Lemma 4.29 and Transitivity for the semantics. The property then follows by induction on \(K\), using Properties 1 and 3, and the fact that if \(\vdash S\ \Delta' \geq \Delta\) then \(\Delta'(X) \equiv \Delta(X)\) for all \(X \in \text{dom}(\Delta)\).

For Property 7, first notice that if \(\Delta \vdash S\ A, B \rightarrow_w C \rightarrow_n D : K'\) and \(A\) is a semantic object for \(\Delta\) and \(K'\) then \(B\) is a semantic object for \(\Delta\) and \(K'\) as well, using Determinism to show that \(\Delta \vdash S\ B \leq T_{K'} : K'\). Furthermore, we can show that if \(\Delta \vdash S\ A, B \rightarrow_w C \rightarrow_n D : K\) and \(\Delta \vdash S\ A E \rightarrow_w F \rightarrow_n G : K'\) then \(\Delta \vdash S\ B E \rightarrow_w F \rightarrow_n G : K'\). The result follows by induction on \(K\), using Thinning and this simple lemma in the II case.

Finally, for Property 8, we first observe that if \(\Delta \vdash S\ K \rightarrow_n K'\) then \(T_K\) is a semantic object for \(\Delta\) and \(K'\), which follows by a simple induction on \(K\). The result follows by induction on \(K\), using Properties 3, 4 and 5, plus Thinning and Conversion of subtyping.

We also need properties corresponding to the model being a Kripke-model:

**Lemma 5.7** (Monotonicity) If \(\vdash S\ \Delta' \geq \Delta\) then:

1. \(A \in \llbracket K \rrbracket \gamma \Delta\) implies \(A \in \llbracket K \rrbracket \gamma \Delta'\).
2. \((A, B) \in \llbracket K \rrbracket \leq \gamma \Delta\) implies \((A, B) \in \llbracket K \rrbracket \leq \gamma \Delta'\).
3. \(\gamma \in \llbracket \Gamma \rrbracket \Delta\) implies \(\gamma \in \llbracket \Gamma \rrbracket \Delta'\).

**Proof:** By induction on \(K\) or \(\Gamma\), using Thinning for the first two and using the first two for the last. 

\[39\]
We also need to account for the dependency, since bounds in kinds include types:

**Lemma 5.8 (Substitution)**

1. If \([K][\gamma_1][X:=B][\gamma_1][\gamma_2][\Delta]\) is defined then \([K][X:=B][\gamma_1][\gamma_2][\Delta]\) is defined and 
   
   \([K][\gamma_1][X:=B][\gamma_1][\gamma_2][\Delta] = [K][X:=B][\gamma_1][\gamma_2][\Delta].\)

2. Suppose that:
   
   - \([\Gamma_1, X\leq B:K, \Gamma_2][\Delta]\) is defined, and
   - \((A[\gamma_1], B[\gamma_1]) \in [K]\leq \gamma_2[\Delta'] for \vdash S \Delta' \geq \Delta and \gamma_1 \in [\Gamma_1][\Delta'].\)
   
   Then:
   
   - \([\Gamma_1, \Gamma_2][X:=A][\Delta]\) is defined, and
   - if \(\gamma_1 \gamma_2 \in [\Gamma_1, \Gamma_2][X:=A][\Delta]\) then \(\gamma_1[X:=A][\gamma_1][\gamma_2] \in [\Gamma_1, X\leq B:K, \Gamma_2][\Delta].\)

**Proof:** Case 1 follows by induction on \(K\), using basic properties of parallel substitution. Case 2 follows by induction on \(\Gamma_2\), using basic properties of parallel substitution and Case 1. 

**Lemma 5.9 (Thinning)**

1. Suppose:
   
   - \([\Gamma_1, \Gamma_2][\Delta]\) is defined, and
   - if \(\vdash S \Delta' \geq \Delta and \gamma_1 \in [\Gamma_1][\Delta']\) then \(A[\gamma_1] \in [*][\gamma_1][\Delta']\).
   
   Then \([\Gamma_1, x:A, \Gamma_2][\Delta]\) is defined and \([\Gamma_1, x:A, \Gamma_2][\Delta] = [\Gamma_1, \Gamma_2][\Delta].\)

2. Suppose:
   
   - \([\Gamma_1, \Gamma_2][\Delta]\) is defined, and
   - if \(\vdash S \Delta' \geq \Delta and \gamma_1 \in [\Gamma_1][\Delta']\) then \(A[\gamma_1] \in [K][\gamma_1][\Delta']\).
   
   Then \([\Gamma_1, X\leq A:K, \Gamma_2][\Delta]\) is defined and if \(\gamma_1[X:=B][\gamma_2] \in [\Gamma_1, X\leq A:K, \Gamma_2][\Delta]\) then \(\gamma_1 \gamma_2 \in [\Gamma_1, \Gamma_2][\Delta].\)

**Proof:** By induction on \(\Gamma_2\), using Lemma 5.5 Case 3 for the type variable case of Case 2.

Finally, we prove a lemma to deal with the rules of context equality.

**Lemma 5.10 (Context Replacement)**

1. Suppose:
   
   - \([\Gamma_1, x:A, \Gamma_2][\Delta]\) is defined, and
   - for any \(\vdash S \Delta' \geq \Delta and \gamma_1 \in [\Gamma_1][\Delta']\) then:
     
     - \(\Delta' \vdash S A[\gamma_1], B[\gamma_1] \rightarrow_n C : [*] for some C,\)
     - \(A[\gamma_1] \in [*][\gamma_1][\Delta'] and B[\gamma_1] \in [*][\gamma_1][\Delta'].\)
   
   Then \([\Gamma_1, x:B, \Gamma_2][\Delta]\) is defined and \([\Gamma_1, x:A, \Gamma_2][\Delta] = [\Gamma_1, x:B, \Gamma_2][\Delta].\)

2. Suppose:
   
   - \([\Gamma_1, X\leq A:K, \Gamma_2][\Delta]\) is defined, and
   - for any \(\vdash S \Delta' \geq \Delta and \gamma_1 \in [\Gamma_1][\Delta']\) then:
\[ \Delta' \vdash S \quad K[\gamma_1], K'[\gamma_1] \rightarrow_n K'' \quad \text{and} \quad \Delta' \vdash S \quad A[\gamma_1], B[\gamma_1] \rightarrow_n C : K'' \quad \text{for some} \quad K'' \quad \text{and} \quad C, \]
\[ [K]_\gamma \Delta' = [K']_\gamma \Delta', \ \text{and} \]
\[ A[\gamma_1] \in [K], \gamma_1 \Delta' \quad \text{and} \quad B[\gamma_1] \in [K'], \gamma_1 \Delta'. \]

Then \([\Gamma_1, X \leq B : K', \Gamma_2] \Delta\) is defined and \([\Gamma_1, X \leq A : K, \Gamma_2] \Delta = [\Gamma_1, X \leq B : K', \Gamma_2] \Delta.\]

**Proof:** By induction on \(\Gamma_2\), using Lemma 5.6 Case 4 for the base case of Case 2. \(\square\)

### 5.3 Soundness

We can now prove soundness. As usual for strong normalization proofs, we first need to prove the more general statement with respect to arbitrary well-behaved substitutions.

**Theorem 5.11** If \(\Gamma \vdash J\) and \(\Delta \vdash S\) ok then \([\Gamma] \Delta\) is defined. Furthermore:

1. If \(\Gamma \vdash K\) and \(\gamma \in [\Gamma] \Delta\) then \([K]_\gamma \Delta\) is defined.
2. If \(\Gamma \vdash K =_\beta K'\) and \(\gamma \in [\Gamma] \Delta\) then \([K]_\gamma \Delta\) and \([K']_\gamma \Delta\) are defined, \([K]_\gamma \Delta = [K']_\gamma \Delta\) and there is a \(K''\) such that \(\Delta \vdash S K[\gamma], K'[\gamma] \rightarrow_n K''\).
3. If \(\Gamma \vdash A : K\) and \(\gamma \in [\Gamma] \Delta\) then \([K]_\gamma \Delta\) is defined and \(A[\gamma] \in [K]_\gamma \Delta\).
4. If \(\Gamma \vdash A =_\beta B : K\) and \(\gamma \in [\Gamma] \Delta\) then \([K]_\gamma \Delta\) is defined, \(A[\gamma]\) and \(B[\gamma]\) are in \([K]_\gamma \Delta\) and there are \(C\) and \(K'\) such that \(\Delta \vdash S K[\gamma] \rightarrow_n K'\) and \(\Delta \vdash S A[\gamma], B[\gamma] \rightarrow_n C : K'\).
5. If \(\Gamma \vdash A \leq B : K\) and \(\gamma \in [\Gamma] \Delta\) then \([K]_\gamma \Delta\) is defined and \((A[\gamma], B[\gamma]) \in [K]_\gamma \Delta\).

**Proof:** By induction on derivations, using the above properties about the interpretation. We consider several cases:

**Thin** By the induction hypothesis \([\Gamma_1, \Gamma_2] \Delta\) and \([\Gamma_1] \Delta\) are defined, and \(\gamma_1 \in [\Gamma_1] \Delta\) implies \(A[\gamma_1] \in [K]_\gamma \Delta\). By Lemma 5.9 Case 2 \([\Gamma_1, X \leq A : K, \Gamma_2] \Delta\) is defined.

Furthermore, \(\gamma_1 [X := B] \gamma_2 \in [\Gamma_1, X \leq A : K, \Gamma_2] \Delta\) implies \(\gamma_1 \gamma_2 \in [\Gamma_1, \Gamma_2] \Delta\) by the same lemma, so we can apply the appropriate induction hypothesis. For example, if \(J \equiv K'\) then \([K']_\gamma \gamma_2 \Delta\) is defined by the induction hypothesis, and \([K']_\gamma \gamma_2 \Delta = [K'']_\gamma \gamma_1 [X := B] \gamma_2 \Delta\) by Lemma 5.5 Case 3.

**Subst** By the induction hypothesis \([\Gamma_1, X \leq B : K, \Gamma_2] \Delta\) and \([\Gamma_1] \Delta\) are defined, and \(\gamma_1 \in [\Gamma_1] \Delta\) implies \((A[\gamma_1], B[\gamma_1]) \in [K]_\gamma \Delta\). By Lemma 5.8 Case 2 \([\Gamma_1, \Gamma_2[X \leftarrow A]] \Delta\) is defined.

Furthermore, \(\gamma_1 \gamma_2 \in [\Gamma_1, \Gamma_2[X \leftarrow A]] \Delta\) implies \(\gamma_1 [X := A[\gamma_1]] \gamma_2 \in [\Gamma_1, X \leq A : K', \Gamma_2] \Delta\), so we can apply the appropriate induction hypothesis. For example, if \(J \equiv K''\) then \([K'']_\gamma \gamma_1 [X := A[\gamma_1]] \gamma_2 \Delta\) is defined by the induction hy-
pothesis, so \([K''[X\leftarrow A]]\gamma_1\gamma_2\Delta\) is defined and \([K'']\gamma_1[X:=A[\gamma_1]]\gamma_2\Delta = [K''[X\leftarrow A]]\gamma_1\gamma_2\Delta\) by Lemma 5.8 Case 1.

**CONTEXT-T-EQ** The induction hypotheses satisfy the premises of Lemma 5.10 Case 2, so \([\Gamma_1, X\leq B:K', \Gamma_2]\Delta\) is defined and

\[[\Gamma_1, X\leq A:K, \Gamma_2]\Delta = [\Gamma_1, X\leq B:K', \Gamma_2]\Delta\]

Hence, if \(\gamma \in [\Gamma_1, X\leq B:K', \Gamma_2]\Delta\) then \(\gamma \in [\Gamma_1, X\leq A:K, \Gamma_2]\Delta\), so we can apply the appropriate induction hypothesis.

**KIND-AGREEMENT** By the induction hypothesis \([\Gamma]\Delta\) is defined.

By the induction hypothesis if \(\gamma \in [\Gamma]\Delta\) then \([K]\gamma\Delta\) is defined.

**C-EMPTY** By definition \([\emptyset]\Delta\) is defined if \(\Delta \vdash _S \emptyset\) ok.

**C-VAR** By the induction hypothesis \([\Gamma]\Delta'\) is defined for any valid context \(\Delta'\), in particular \(\vdash _S \Delta' \geq \Delta\). If \(\gamma \in [\Gamma]\Delta'\), then \(A[\gamma] \in [K]_{\Delta}\Delta'\) by the induction hypothesis. Furthermore, \(x \notin \text{dom}(\Gamma)\), so \([\Gamma, x:A]\Delta\) is defined.

**C-TVAR** By the induction hypothesis \([\Gamma]\Delta'\) is defined for \(\vdash _S \Delta' \geq \Delta\). If \(\gamma \in [\Gamma]\Delta'\), then \(A[\gamma] \in [K]_{\Delta}\Delta'\) by the induction hypothesis, and \(x \notin \text{dom}(\Gamma)\), so \([\Gamma, x:A]\Delta\) is defined.

**T-TOP** By the induction hypothesis \([\Gamma]\Delta\) is defined.

Suppose \(\gamma \in [\Gamma]\Delta\). We know \(\Delta \vdash _S \emptyset\) ok by Lemma 5.5 Case 2, so \([\emptyset]\gamma\Delta\) is defined. Furthermore, \(\Delta \vdash _S \top_\ast \rightarrow_n \top_\ast : \ast\) by ST-Top, and hence \(\Delta \vdash _S \top_\ast \leq \top_\ast : \ast\) by SWS-Top and SS-Inc. Hence \(\top_\ast\) is a semantic object for \(\Delta\) and \(\ast\), and so \(\top_\ast \in [\ast]\gamma\Delta\).

**T-TVAR** By the induction hypothesis \([\Gamma_1, X\leq A : K, \Gamma_2]\Delta\) is defined.

Suppose \(\gamma \in [\Gamma]\Delta\). Then \(\gamma = \gamma_1[X:=B]\gamma_2\), with \(\gamma_1 \in [\Gamma_1]\Delta\) and \((B, A[\gamma_1]) \in [K][\gamma_1]\Delta\), by definition of \([\Gamma]\Delta\). Hence \(B \in [K][\gamma_1]\Delta\) by Lemma 5.6 Case 3, and \(X[\gamma] \equiv B \in [K][\gamma\Delta]\) by Lemma 5.5 Case 3.

**T-TABS** By the induction hypothesis \([\Gamma, X\leq A_1,K_1]\Delta\) is defined, so by definition of the interpretation \([\Gamma]\Delta'\) is defined and \(A_1[\gamma] \in [K_1][\gamma\Delta']\) for \(\gamma \in [\Gamma]\Delta'\), for any \(\vdash _S \Delta' \geq \Delta\).

Suppose \(\gamma \in [\Gamma]\Delta\). We want to show \((\Lambda X \leq A_1 : K_1 . A_2)[\gamma] \in [\Pi X \leq A_1 : K_1, K_2] : \gamma\Delta\).

We have to show two conditions:

- The first condition is that \(\Delta \vdash _S (\Pi X \leq A_1 : K_1, K_2)[\gamma] \rightarrow_n K'\) and \((\Lambda X \leq A_1 : K_1 . A_2)[\gamma]\) is a semantic object for \(\Delta\) and \(K'\). By Lemma 5.5 Case 1 \(\Delta \vdash _S K_1[\gamma] \rightarrow_n K'_1\), and by Lemma 5.6 Case 1 \(\Delta \vdash _S A_1[\gamma] \rightarrow_n A'_1 : K'_1\). Hence \(\Delta, Y \leq A_1[\gamma] : K_1[\gamma] \vdash _S \emptyset\) ok for \(Y\) fresh in \(\Delta\).

- By Lemma 5.7 Case 3 \(\gamma \in [\Gamma]\Delta\) implies \(\gamma \in [\Gamma]\Delta\), \(Y \leq A_1[\gamma] : K_1[\gamma]\). By Lemma 5.6 Case 6 \((Y, A_1[\gamma]) \in [K_1][\gamma\Delta], Y \leq A_1[\gamma] : K_1[\gamma]\). Hence \(\gamma[X:=Y] \in [\Gamma, X \leq A_1 : K_1]\Delta, Y \leq A_1[\gamma] : K_1[\gamma]\)

so by the induction hypothesis \(A_2[\gamma[X:=Y]] \in [K_2][\gamma[X:=Y]\Delta, Y \leq A_1[\gamma] : K_1[\gamma]]\).
Hence

\[ \Delta, Y \leq A_1[\gamma]: K_1[\gamma] \vdash_s K_2[\gamma|X:=Y]| \rightarrow_n K'_2 \quad \text{by Lemma 5.5 Case 1}, \]
\[ \Delta, Y \leq A_1[\gamma]: K_1[\gamma] \vdash_s A_2[\gamma|X:=Y]| \leq T_{K'_2}: K'_2 \quad \text{by Lemma 5.6 Case 1}, \]
\[ \Delta, Y \leq A_1[\gamma]: K_1[\gamma] \vdash_s A_2[\gamma|X:=Y]| \rightarrow_n A_2': K'_2 \quad \text{by inversion, for some } A_2'. \]

Hence,

\[ \Delta \vdash_s (\Pi X \leq A_1: K_1).K_2)[\gamma] \rightarrow_n \Pi Y \leq A_1'[K_1'].K'_2 \quad \text{by SK-II}, \]
\[ \Delta \vdash_s (\Lambda X \leq A_1: K_1).A_2)[\gamma] \equiv \Lambda Y \leq A_1[\gamma]: K_1[\gamma].A_2[\gamma|X:=Y]| \rightarrow_n \Lambda Y \leq A_1'[K_1'].A_2'[\gamma]| \Pi Y \leq A_1'[K_1'].K'_2 \quad \text{by ST-TABS}, \]
\[ \Delta \vdash_s (\Lambda X \leq A_1: K_1).A_2)[\gamma] \leq \Lambda Y \leq A_1'[K_1'].A_2'[\gamma]| \Pi Y \leq A_1'[K_1'].K'_2 \quad \text{by WSS-TABS}, \]

where the last line follows using Adequacy and Subject Reduction for the well-typedness of the right-hand side. Hence, \((\Lambda X \leq A_1: K_1).A_2)[\gamma]\) is a semantic object for \(\Delta\) and \(\Pi Y \leq A_1'[K_1'].K'_2\).

- The second condition is that \((\Lambda X \leq A_1: K_1).A_2)[\gamma]| B \in [K_2].\gamma|X:=B|\Delta'|, for \(\vdash_s \Delta' \geq \Delta\) and \((B, A_1[\gamma]) \in [K_1]| \leq \Delta'\). First, by Lemma 5.7 Case 3 \(\gamma \in [\Gamma]|\Delta\) implies \(\gamma \in [\Gamma]|\Delta'\), and \((B, A_1[\gamma]) \in [K_1]| \leq \Delta'\) implies \(\gamma|X:=B| \in [\Gamma', X \leq A_1: K_1]| \Delta'\) by definition of the interpretation. Hence by the induction hypothesis \(A_2[\gamma|X:=B]| \in [K_2].\gamma|X:=B|\Delta'\), and so by Lemma 5.6 Case 1 \(\Delta' \vdash_s A_2[\gamma|X:=B]| \rightarrow_w C : K_2'|\), where \(\Delta' \vdash_s K_2[\gamma|X:=B]| \rightarrow_n K_2''\). Then \(\Delta \vdash_s (\Lambda X \leq A_1: K_1).A_2)[\gamma]| : \Pi Y \leq A_1'[K_1'].K'_2\) implies \(\Delta' \vdash_s (\Lambda X \leq A_1: K_1).A_2)[\gamma]| : \Pi Y \leq A_1'[K_1'].K'_2\) by Thinning, and \(\Delta' \vdash_s B \leq A_1[\gamma]| : K_1'\) by Lemma 5.6 Case 2. Hence

\[ \Delta' \vdash_s (\Lambda X \leq A_1: K_1).A_2)[\gamma]|(B) \equiv (\Lambda Y \leq A_1[\gamma]: K_1[\gamma].A_2[\gamma|X:=Y]|)(B) \rightarrow_w C : K_2' \]

by ST-Beta, because \(A_2[\gamma|X:=Y]|(Y \leftarrow B) \equiv A_2[\gamma|X:=B]|\) where \(Y\) can be chosen to be fresh in \(\Delta'\). Hence, by Lemma 5.6 Case 7

\[ (\Lambda X \leq A_1: K_1).A_2)[\gamma]|(B) \in [K_2].\gamma|X:=B|\Delta' \]

Hence, \((\Lambda X \leq A_1: K_1).A_2)[\gamma]| \in [\Pi X \leq A_1: K_1].K_2]|.\gamma|\Delta.\)

**T-TAPP** By the induction hypothesis \([\Gamma]|\Delta\) is defined.

Suppose \(\gamma \in [\Gamma]|\Delta\). Then \(A[\gamma]| \in [\Pi X \leq B: K_1].K_2]|.\gamma|\Delta\) and \((C[\gamma], B[\gamma]) \in [K_1]| \leq \gamma|\Delta\). Clearly \(\vdash_s \Delta \geq \Delta\), since \(\Delta \vdash_s \Pi|B\) by Lemma 5.5 Case 2, so by definition of \([\Pi X \leq B: K_1].K_2]|.\gamma|\Delta:\)

\[ (A C)[\gamma] \equiv (A[\gamma]|(C[\gamma]) \in [K_2].\gamma|X:=C[\gamma]|) = [K_2|X \leftarrow C]|.\gamma|\Delta \]

where the last equality follows by Lemma 5.8 Case 2.

**T-EQ-TAPP** By the induction hypothesis \([\Gamma]|\Delta\) is defined.
Suppose $\gamma \in [\Gamma] \Delta$. By the induction hypothesis $A[\gamma], B[\gamma] \in [\Pi X \leq E: K_1, K_2][\gamma]$, $\Delta \vdash_S A[\gamma], B[\gamma] \rightarrow_n a' : \Pi X \leq E': K'_1, K'_2$ with $\Delta \vdash_S (\Pi X \leq E: K_1, K_2)[\gamma] \rightarrow_n \Pi X \leq E' : K'_1, K'_2$, and also $C[\gamma], D[\gamma] \in [K_1], \Delta \vdash_S C[\gamma], D[\gamma] \rightarrow_n C' : K'_1$, and finally $C[\gamma], E[\gamma]) \in [K_1] \ll \gamma \Delta$. Hence,

$$(AC)[\gamma] \equiv (A[\gamma])(C[\gamma]) \in [K_2], \gamma[X := C[\gamma]] \Delta = [K_2[X \leftarrow C]] \Delta$$

$$(BD)[\gamma] \equiv (B[\gamma])(D[\gamma]) \in [K_2], \gamma[X := D[\gamma]] \Delta = [K_2[X \leftarrow D]] \Delta$$

Using Adequacy, Subject Reduction and Determinacy we conclude that

$$\Delta \vdash_S (AC)[\gamma], (BD)[\gamma] \rightarrow_n F : K'_2$$

where $\Delta \vdash_S K_2[X \leftarrow C] \rightarrow_n K'_2$. Finally, $\Delta \vdash_S \gamma[X := C[\gamma]], \gamma[X := D[\gamma]] \rightarrow_n \gamma'$, so by Lemma 5.5 Case 4:

$$(BD)[\gamma] \in [K_2], \gamma[X := C[\gamma]] \Delta = [K_2[X \leftarrow C]] \Delta$$

S-TOP By the induction hypothesis $[\Gamma] \Delta$ is defined.

Suppose $\gamma \in [\Gamma] \Delta$. Then $A[\gamma] \in [K], \gamma \Delta$, so by Lemma 5.6 Case 8 $A[\gamma] \rightarrow_T K[\gamma] \gamma \Delta$, and $T_K[\gamma] \equiv T_K[\gamma]$.

S-TRANS By the induction hypothesis $[\Gamma] \Delta$ is defined.

Suppose $\gamma \in [\Gamma] \Delta$. By the induction hypothesis $(A[\gamma], B[\gamma]) \in [K] \ll \gamma \Delta$ and $(B[\gamma], C[\gamma]) \in [K] \ll \gamma \Delta$. Hence by Lemma 5.6 Case 5 $(A[\gamma], C[\gamma]) \in [K] \ll \gamma \Delta$. □

**Lemma 5.12** If $[\Gamma]$ is defined then $[\Gamma][\Gamma]$ is also defined and $id_{\Gamma} \in [\Gamma][\Gamma]$, where $id_{\Gamma}$ is the identity substitution on $\Gamma$.

**Proof:** By induction on $\Gamma$, using Lemma 5.6 Case 6 and Monotonicity. □

**Corollary 5.13 (Soundness)**

1. If $\Gamma \vdash ok$ then $\Gamma \vdash_S ok$.
2. If $\Gamma \vdash K$ then there is a $K'$ such that $\Gamma \vdash_S K \rightarrow_n K'$.
3. If $\Gamma \vdash K =_S K'$ then there is a $K''$ such that $\Gamma \vdash_S K \rightarrow_n K''$ and $\Gamma \vdash_S K' \rightarrow_n K''$.
4. If $\Gamma \vdash A : K$ then there are $K'$, $B$ and $C$ such that $\Gamma \vdash_S K \rightarrow_n K'$ and $\Gamma \vdash_S A \rightarrow_S B \rightarrow_S C : K'$.
5. If $\Gamma \vdash A =_S B : K$ then there are $C$ and $K'$ such that $\Gamma \vdash_S K \rightarrow_n K'$, $\Gamma \vdash_S A \rightarrow_S C : K'$ and $\Gamma \vdash_S B \rightarrow_S C : K'$.
6. If $\Gamma \vdash A \leq_S B : K$ then there is a $K'$ such that $\Gamma \vdash_S K \rightarrow_n K'$ and $\Gamma \vdash_S A \leq B : K'$.

**Proof:** By C-EMPTY $\emptyset \vdash ok$, so by Theorem 5.11 $[\Gamma] \emptyset$ is defined. Hence, $[\Gamma][\Gamma]$ is defined and $id_{\Gamma} \in [\Gamma][\Gamma]$ by Lemma 5.12, and by Lemma 5.5 Case 2 we know $\Gamma \vdash_S ok$. The result follows by Theorem 5.11. We show here Case 4.
By Theorem 5.11 Case 3 \([K][\text{id} \Gamma] \) is defined, and by Lemma 5.5 Case 1 \(\Gamma \vdash_{S} K \rightarrow_n K'\) for some \(K'\), since \(K[\text{id} \Gamma] = K\). By Theorem 5.11 Case 3, \(A \equiv A[\text{id} \Gamma] \in [K][\text{id} \Gamma] \) and by Lemma 5.6 Case 1 \(A\) is a semantic object for \(\Gamma\) and \(K'\). By the definition of semantic object and inversion there exist \(B\) and \(C\) such that \(\Gamma \vdash_{S} A \rightarrow_{w} B \rightarrow_{w} C : K'\).

\[\square\]

### 5.4 Consequences of Soundness

We can use Soundness, Corollary 5.13, and Completeness, Proposition 4.12, to transfer the metatheoretic results from the typed operational semantics to the original presentation.

**Lemma 5.14 (Admissibility of Structural Rules)** The rules in Section 2.3 are admissible for the system \(\Gamma \vdash \neg J\).

**Proof:** Suppose we have a derivation of \(\Gamma \vdash J\), for example \(\Gamma \vdash A : K\), which is then a derivation of \(\Gamma \vdash \neg J\) with uses of the structural rules in Section 2.3. By Soundness we know that there are \(B\) and \(K'\) such that \(\Gamma \vdash_{S} A \rightarrow_{w} B : K'\) and \(\Gamma \vdash_{S} K \rightarrow_{n} K'\). By Completeness \(\Gamma \vdash \neg A : K'\) and \(\Gamma \vdash \neg K =_{\beta} K'\), so by T-CONV \(\Gamma \vdash \neg A : K\).

\[\square\]

**Lemma 5.15 (Strong Normalization)** If \(\Gamma \vdash A : K\) then \(A\) is strongly normalizing.

**Proof:** By Soundness and Strong Normalization (Lemma 4.20).

\[\square\]

**Lemma 5.16 (Subject Reduction for \(\rightarrow_{\beta_2}\))**

- If \(\Gamma \vdash \text{ok}\) and \(\Gamma \rightarrow_{\beta_2} \Gamma'\) then \(\Gamma' \vdash \text{ok}\).
- If \(\Gamma \vdash K\) and \(K \rightarrow_{\beta_2} K'\) then \(\Gamma \vdash K'\) and \(\Gamma \vdash K =_{\beta} K'\).
- If \(\Gamma \vdash A : K\) and \(A \rightarrow_{\beta_2} B\) then \(\Gamma \vdash B : K\) and \(\Gamma \vdash A =_{\beta} B : K\).
- If \(\Gamma \vdash A \leq B : K\) and \(A \rightarrow_{\beta_2} C\) then \(\Gamma \vdash C \leq B : K\), or if \(B \rightarrow_{\beta_2} C\) then \(\Gamma \vdash A \leq C : K\).

**Proof:** By Soundness, Subject Reduction (Corollary 4.17), and Completeness.

\[\square\]

**Proposition 5.17 (Generation for Subtyping)**

1. If \(\Gamma \vdash (A_1 \rightarrow A_2) \leq (B_1 \rightarrow B_2) : \ast\) then \(\Gamma \vdash B_1 \leq A_1 : \ast\) and \(\Gamma \vdash A_2 \leq B_2 : \ast\).
2. If \(\Gamma \vdash (\forall X \leq A_1 : K_A . A_2) \leq (\forall X \leq B_1 : K_B . B_2) : \ast\) then \(\Gamma \vdash K_A =_{\beta} K_B\), \(\Gamma \vdash A_1 =_{\beta} B_1 : K_A\), and \(\Gamma, X \leq A_1 : K_A \vdash A_2 \leq B_2 : \ast\).

**Proof:**

45
(1) By Soundness \( \Gamma \vdash_s A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2 : * \). Since the semantic presentation is deterministic the latter must have been obtained by SS-INC and SWS-ARROW from \( \Gamma \vdash_s B_1 \leq A_1 : * \) and \( \Gamma \vdash_s A_2 \leq B_2 : * \). Then, by Completeness, \( \Gamma \vdash B_1 \leq A_1 : * \) and \( \Gamma \vdash A_2 \leq B_2 : * \).

(2) By Soundness \( \Gamma \vdash_s \forall X \leq A_1 : K_A \rightarrow A_2 \leq \forall X \leq B_1 : K_B \rightarrow B_2 : * \). Since the semantic presentation is deterministic the latter must have been obtained by SS-INC and SWS-ALL from \( \Gamma, X \leq A_1 : K_A \vdash_s A_2 \leq B_2 : * \), \( \Gamma \vdash_s A_1, B_1 \rightarrow_n C : K'' \), and \( \Gamma \vdash_s K_A, K_B \rightarrow_n K'' \). By Completeness \( \Gamma, X \leq A_1 : K_A \vdash A_2 \leq B_2 : * \), and also \( \Gamma \vdash A_1 = \beta C : K'' \), \( \Gamma \vdash B_1 = \beta C : K'' \), and \( \Gamma \vdash K_A = \beta K'' \). Hence by T-Eq-Sym, T-Eq-Trans, K-Eq-Sym, K-Eq-Trans, it follows that \( \Gamma \vdash K_A = \beta K_B \), and also that \( \Gamma \vdash A_1 = \beta B_1 : K'' \), and \( \Gamma \vdash K'' = \beta K_A \). Finally, by T-Eq-Conv, \( \Gamma \vdash A_1 = \beta B_1 : K_A \). \( \square \)

**Proposition 5.18 (Generation for Typing)**

(1) If \( \Gamma \vdash \lambda x:A_1.M : A \) then there exists an \( A_2 \) such that \( \Gamma, x:A_1 \vdash M : A_2 \) and \( \Gamma \vdash A_1 \rightarrow A_2 \leq A : * \).

(2) If \( \Gamma \vdash \lambda X:A_1.K.M : A \) then there exists an \( A_2 \) such that \( \Gamma, X \leq A_1 : K \vdash M : A_2 \) and \( \Gamma \vdash \forall X \leq A_1 : K.A_2 \leq A : * \).

**Proof:** Each case follows by induction on the derivation of the antecedent. \( \square \)

**Lemma 5.19 (Agreement)**

(1) If \( \Gamma_1, x:A, \Gamma_2 \vdash \text{ok} \) then \( \Gamma_1 \vdash A : * \).

(2) If \( \Gamma_1, X \leq A : K, \Gamma_2 \vdash \text{ok} \) then \( \Gamma_1 \vdash A : K \).

(3) If \( \Gamma \vdash M : A \) then \( \Gamma \vdash A : * \).

**Proof:** We use Lemma 4.6, Soundness and Completeness. \( \square \)

**Lemma 5.20 (Upper Bound)**

(1) If \( \Gamma \vdash X(A_1, \ldots, A_m) : K \) then \( \Gamma \vdash \Gamma(X)(A_1, \ldots, A_m) : K \).

(2) If \( \Gamma \vdash X(A_1, \ldots, A_m) : K \) then \( \Gamma \vdash X(A_1, \ldots, A_m) \leq \Gamma(X)(A_1, \ldots, A_m) : K \).

**Proof:**

(1) By induction on \( m \), using Soundness, Completeness, Adequacy, Determinacy, and Subject Reduction.

(2) By the previous item, \( \Gamma \vdash \Gamma(X)(A_1, \ldots, A_m) : K \). By Soundness, there exists \( K' \) such that \( \Gamma \vdash_s \Gamma(X)(A_1, \ldots, A_m) : K' \) and \( \Gamma \vdash_s K \rightarrow_n K' \). By Lemma 4.29, \( \Gamma \vdash X(A_1, \ldots, A_m) \leq \Gamma(X)(A_1, \ldots, A_m) : K' \), and, by Completeness, \( \Gamma \vdash X(A_1, \ldots, A_m) \leq \Gamma(X)(A_1, \ldots, A_m) : K' \) and
\[ \Gamma \vdash K =_\beta K'. \] Finally, by K-Eq-SYM and S-K-_CONV, it follows that \[ \Gamma \vdash X(A_1, \ldots, A_m) \leq \Gamma(X)(A_1, \ldots, A_m) : K. \] \[ \square \]

6 Subject Reduction

In this section we show Subject Reduction for terms. We first need to show some basic properties of the system with respect to judgements with term variables.

**Lemma 6.1** If \( \Gamma \vdash ok \) then all variables in \( \text{dom}(\Gamma) \) are distinct.

**Proof:** By Soundness, Lemma 4.3 and Completeness. \[ \square \]

**Lemma 6.2** If \( \Gamma \vdash J \) and \( \Gamma_0 \) is a prefix of \( \Gamma \), then \( \Gamma_0 \vdash ok \).

**Proof:** By Soundness, Lemma 4.6 and Completeness. \[ \square \]

**Lemma 6.3 (Strengthening)** If \( \Gamma_1, y:C, \Gamma_2 \vdash J \) and \( y \not\in \text{FV}(J) \) then \( \Gamma_1, \Gamma_2 \vdash J \).

**Proof:** By induction on the derivation of \( \Gamma_1, y:C, \Gamma_2 \vdash J \). Most cases follow by the induction hypothesis and the corresponding rule, and C-VAR uses Lemma 6.2. \[ \square \]

**Lemma 6.4** If \( Y \not\in \text{FTV}(C) \) then \( B[X \leftarrow C][Y \leftarrow A[X \leftarrow C]] \equiv B[Y \leftarrow A][X \leftarrow C] \).

**Proof:** By induction on the structure of \( B \). \[ \square \]

**Lemma 6.5 (Term Substitution)**

1. If \( \Gamma_1, x:A, \Gamma_2 \vdash M : B \) and \( \Gamma_1 \vdash N : A \) then \( \Gamma_1, \Gamma_2 \vdash M[x \leftarrow N] : B \).
2. If \( \Gamma_1, X \leq A : K, \Gamma_2 \vdash M : B \) and \( \Gamma_1 \vdash C \leq A \) then \( \Gamma_1, \Gamma_2[X \leftarrow C] \vdash M[X \leftarrow C] : B[X \leftarrow C] \).

**Proof:**

1. By induction on derivations, where for T-VAR we use the structural rule THIN and Strengthening (Lemma 6.3), and for T-ABS we use Lemmas 6.2 and 6.1.
2. By induction on derivations, using Lemmas 6.2 and 6.1 in the case T-TABS; using Lemma 6.4 and the rule SUBST in the case T-TAPP; and using the rule SUBST in the case T-SUB. \[ \square \]

**Proposition 6.6 (\( \rightarrow_{\beta_1} \) Subject Reduction for Terms)**
If $\Gamma \vdash M : A$ and $M \rightarrow_{\beta_1} M'$ then $\Gamma \vdash M' : A$.

**Proof:** By induction on the derivation of $\Gamma \vdash M : A$.

**T-Var** Vacuously true.

**T-Abs** By the induction hypothesis and T-Abs.

**T-App** Let $M \equiv N \cdot P$. Then we are given that $\Gamma \vdash N : A_1 \rightarrow A$ and $\Gamma \vdash P : A_1$.

There are 3 cases to consider. If $N \rightarrow_{\beta_1} N'$ or $P \rightarrow_{\beta_1} P'$ then the result follows by the induction hypothesis and T-App. The interesting case is when $N \equiv \lambda x : B_1.N'$ and $M \rightarrow_{\beta_1} N'[x \leftarrow P]$. By Generation for Typing (Proposition 5.18), $\Gamma, x : B_1 \vdash N' : B_2$ and $\Gamma \vdash B_1 \rightarrow B_2 \leq A_1 \rightarrow A$, for some $B_2$. By Generation for Subtyping (Proposition 5.17) $\Gamma \vdash A_1 \leq B_1 : *$ and $\Gamma \vdash B_2 \leq A : *$. By T-Sub, $\Gamma \vdash P : B_1$, and, by Substitution (Lemma 6.5), $\Gamma \vdash N'[x \leftarrow P] : B_2$. Finally, by T-Sub, $\Gamma \vdash N'[x \leftarrow P] : A$.

**T-Tabs** By the induction hypothesis and T-Tabs.

**T-Tapp** Let $M \equiv N \cdot B$. We are given that $\Gamma \vdash N : \forall X \leq A_1 : K_A \cdot A_2$, $\Gamma \vdash B \leq A_1 : K_A$, and $A \equiv A_2[X \leftarrow B]$. There are two cases to consider. If $N \rightarrow_{\beta_1} N'$ then the result follows by the induction hypothesis and T-Tapp. Otherwise, $N \equiv \lambda X \leq B_1 : K_B \cdot N'$ and $M' \equiv N'[X \leftarrow B]$. By Generation for Typing (Proposition 5.18), $\Gamma, X \leq B_1 : K_B \vdash N' : B_2$, and $\Gamma \vdash \forall X \leq B_1 : K_B \cdot B_2 \leq \forall X \leq A_1 : K_A \cdot A_2 : *$, for some $B_2$. By Generation for Subtyping (Proposition 5.17), $\Gamma, X \leq A_1 : K_A \vdash B_2 \leq A_2 : *, \Gamma \vdash A_1 =_{\beta} B_1 : K_A$, and $\Gamma \vdash K_A =_{\beta} K_B$. By S-Conv, $\Gamma \vdash A_1 \leq B_1 : K_A$, and, by S-Trans, $\Gamma \vdash B \leq B_1 : K_A$. By Substitution (Lemma 6.5), $\Gamma \vdash N'[X \leftarrow B] : B_2[X \leftarrow B]$. By Subst, $\Gamma \vdash B_2[X \leftarrow B] \leq A_2[X \leftarrow B] : *$. Finally, by T-Sub $\Gamma \vdash N'[X \leftarrow B] : A_2[X \leftarrow B] : *$.

**T-Sub** By the induction hypothesis and T-Sub.

**Lemma 6.7** If $\Gamma \vdash M : A$ and $\Gamma \rightarrow_{\beta_2} \Gamma'$ then $\Gamma' \vdash M : A$.

**Proof:** By induction on derivations using Lemma 5.16.

**Lemma 6.8** If $C \rightarrow_{\beta_2} C'$ then $B[X \leftarrow C] \rightarrow_{\beta_2} B[X \leftarrow C']$.

**Proposition 6.9** ($\rightarrow_{\beta_2}$ Subject Reduction for Terms)

If $\Gamma \vdash M : A$ and $M \rightarrow_{\beta_2} M'$ then $\Gamma \vdash M' : A$.

**Proof:** By induction on derivations.

**T-Var** Vacuously true.

**T-Abs** There are two cases to consider.

1. $A \rightarrow_{\beta_2} A'$. By Lemma 6.7, $\Gamma, x : A' \vdash M : B$, by T-Abs, $\Gamma \vdash \lambda x : A'. M : A' \rightarrow B$.

By Lemma 5.19 Case 1 and Lemma 6.2 we conclude $\Gamma \vdash A : *$ from $\Gamma, x : A \vdash M : B$. By Lemma 5.16, $\Gamma \vdash A =_{\beta} A' : *$. By Lemma 5.19 Case 3, $\Gamma, x : A' \vdash B : *$, by Strengthening (Lemma 6.3), $\Gamma \vdash B : *$.
by T-EQ-REFL, $\Gamma \vdash B =_\beta B : \star$, by T-EQ-ARROW and T-EQ-SYM,
$\Gamma \vdash A' \rightarrow B =_\beta A \rightarrow B : \star$. Then, by S-Conv, $\Gamma \vdash A' \rightarrow B \leq A \rightarrow B : \star$,
and by T-SUB $\Gamma \vdash \lambda x:A'.M : A \rightarrow B$.

(2) $M \rightarrow^* M'$. By the induction hypothesis and T-ABS.

T-APP There are two cases to consider, either $M \rightarrow^* M'$ or $N \rightarrow^* N'$. Both cases follow by the induction hypothesis and T-APP.

T-TABS There are three cases to consider.

(1) If $M \rightarrow^* M'$, the result follows by the induction hypothesis and T-TABS.

(2) If $A \rightarrow^* A'$, the proof is similar to T-ABS case 1.

(3) If $K \rightarrow^* K'$, by Lemma 6.7, $\Gamma, X \leq A : K' \vdash M : B$, by T-TABS,
$\Gamma \vdash \lambda X \leq A : K'.M : \forall X \leq A : K'.B$. From $\Gamma, X \leq A : K \vdash M : B$, by
Lemma 6.2 and Lemma 5.19 Case 2, $\Gamma \vdash A : K$, and, by the rule KIND-AGREEMENT, $\Gamma \vdash K$. By Lemma 5.16, $\Gamma \vdash K =_\beta K'$, by
T-EQ-REFL, $\Gamma \vdash A =_\beta A : K$. With a similar argument to that
used in the T-ABS case we prove that $\Gamma \vdash B =_\beta B : \star$. Now, by
T-EQ-SYM and T-EQ-ALL, $\Gamma \vdash \forall X \leq A : K'.B =_\beta \forall X \leq A : K.B : \star$, by
S-Conv, $\Gamma \vdash \forall X \leq A : K'.B \leq \forall X \leq A : K.B : \star$. Finally, by T-SUB,
$\Gamma \vdash \lambda X \leq A : K'.M : \forall X \leq A : K.B$.

T-TAPP There are two cases to consider.

(1) If $M \rightarrow^* M'$ then the result follows by the induction hypothesis and T-TAPP.

(2) If $C \rightarrow^* C'$, by Lemma 5.16, $\Gamma \vdash C' \leq A : K$, by T-TAPP, $\Gamma \vdash M C' : B[C' \leftarrow X]$, by Lemma 5.19 Case 3, $\Gamma \vdash B[C' \leftarrow X] =_\beta B[X \leftarrow C] : \star$, by Lemma 6.8 and Lemma 5.16, $\Gamma \vdash B[C' \leftarrow X] =_\beta B[X \leftarrow C] : \star$, and finally by T-EQ-SYM and T-SUB, $\Gamma \vdash M C' : B[X \leftarrow C]$.

T-SUB By the induction hypothesis and T-SUB.

PROPOSITION 6.10 (→∗ Subject Reduction for Terms)

If $\Gamma \vdash M : A$ and $M \rightarrow^* M'$ then $\Gamma \vdash M' : A$.

PROOF: The proof is by induction on the definition of $\rightarrow^*$. The reflexive case
is immediate, the cases for $\rightarrow^*_1$ and $\rightarrow^*_2$ follow by Propositions 6.6 and 6.9 respectively, and the transitivity case follows by the induction hypothesis. □

7 An Algorithmic Presentation

From the rules for subtyping in the typed operational semantics we extract an
algorithm that computes the subtyping relation on weak-head normal forms, ignoring kind information and well-formation of contexts. The reductions to
weak-head normal form and normal form are then untyped calculations defined
as follows:

\[
\begin{align*}
X & \rightarrow_n X \\
T_\ast & \rightarrow_n T_\ast \\
A B & \rightarrow_n C B & & \text{if } A \rightarrow_n C, \ B \text{ normal and } A \text{ a variable or application} \\
A_1 \rightarrow A_2 & \rightarrow_n B_1 \rightarrow B_2 & & \text{if } A_i \rightarrow_n B_i \\
\forall X \leq A_1; K.A_2 & \rightarrow_n \forall X \leq B_1; K'.B_2 & & \text{if } A_i \rightarrow_n B_i \text{ and } K \rightarrow_n K' \\
\Lambda X \leq A_1; K.A_2 & \rightarrow_n \Lambda X \leq B_1; K'.B_2 & & \text{if } A_i \rightarrow_n B_i \text{ and } K \rightarrow_n K' \\
A & \rightarrow_n C & & \text{if } A \rightarrow_w B \text{ and } B \rightarrow_n C \\
A & \rightarrow_w A & & \text{if } A \text{ is weak-head normal and not an application} \\
A B & \rightarrow_w C D & & \text{if } A \rightarrow_w C, \ B \rightarrow_n D \text{ and } C \text{ a variable or application} \\
A B & \rightarrow_w E & & \text{if } A \rightarrow_w \Lambda X \leq C; K.D \text{ and } D[X \leftarrow B] \rightarrow_w E
\end{align*}
\]

with the obvious extension to kinds and contexts. If \( U \rightarrow_n V \), we define \( \text{nf}(U) = V \).

**Lemma 7.1**

(1) If \( K \rightarrow_n K' \) then \( K \rightarrow_{\beta} K' \).

(2) If \( A \rightarrow_n B \) or \( A \rightarrow_w B \) then \( A \rightarrow_{\beta} B \).

**Lemma 7.2** If \( A \rightarrow_n B \) then \( A \) is weak-head normal and \( B \rightarrow_n B \).

**Lemma 7.3**

(1) If \( A \rightarrow_n B \) and \( A \rightarrow_n C \) then \( B \equiv C \).

(2) If \( A \rightarrow_w B \) and \( A \rightarrow_w C \) then \( B \equiv C \).

**Lemma 7.4**

(1) If \( \Gamma \vdash_S K \rightarrow_n K' \) then \( K \rightarrow_n K' \).

(2) If \( \Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K \) then \( A \rightarrow_w B, B \rightarrow_n C \) and \( A \rightarrow_n C \).

**Lemma 7.5**

(1) If \( \Gamma \vdash K, K' \) and \( \text{nf}(K) = \text{nf}(K') \) then \( \Gamma \vdash K \equiv_{\beta} K' \).

(2) If \( \Gamma \vdash A, B : K \) and \( \text{nf}(A) = \text{nf}(B) \) then \( \Gamma \vdash A \equiv_{\beta} B : K \).

The algorithm to compute the subtyping relation is defined by the following rules.

### 7.1 Algorithmic Subtyping

\[
\begin{array}{cccc}
A & \rightarrow_w C & & B \rightarrow_w D & \Gamma \vdash A \ C \leq_w D & \Gamma \vdash A \leq B
\end{array}
\]

(AS-INC)
This rule reduces the arguments to weak-head normal form and then invokes the weak-head subtyping algorithm defined as follows.

7.2 Algorithmic Weak-Head Subtyping

\[
\begin{align*}
\Gamma \vdash_A A \leq_W T & \quad \text{HV}(A) \text{ undefined} & (\text{AWS-TOP}) \\
\Gamma(X) \rightarrow_e B & \quad B(A_1, \ldots, A_m) \rightarrow_w E \\
\Gamma \vdash_A E \leq_W C & \quad C \neq X(A_1, \ldots, A_m) \\
\frac{}{\Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W C} & (\text{AWS-TAPP}) \\
\Gamma \vdash_A A_1 \leq_A A_2 \leq_B B_1 \rightarrow B_2 & (\text{AWS-ARROW})
\end{align*}
\]

\[
\begin{align*}
\Gamma, \ X \leq_A : K & \vdash_A A_2 \leq_B B_2 & \text{nf}(A_1) \equiv \text{nf}(B_1) & \text{nf}(K) \equiv \text{nf}(K') & (\text{AWS-ALL}) \\
\Gamma, \ X \leq_A : K & \vdash_A A_2 \leq_B B_2 & \text{nf}(A_1) \equiv \text{nf}(B_1) & \text{nf}(K_1) \equiv \text{nf}(K_1') & (\text{AWS-TABS})
\end{align*}
\]

Our aim is to prove that this algorithm is sound with respect to the original system on well-formed types: if \( \Gamma \vdash_A A \leq B \) and \( \Gamma \vdash_A B \leq K \) then \( \Gamma \vdash_A A \leq B : K \). The problem we encounter is that the original system uses subtyping to check \( \Gamma \vdash_A B : K \). Therefore we need algorithmic versions of the judgements involved in proving well-kindness.

The algorithmic rules for the other judgements are modifications of the typed operational semantics, but there is only one rule to assign a kind to a type application. In the rules for Kind and Type Formation the assumptions of the form \( \Gamma \vdash \text{ok} \) are dropped. The rules AK-Π, AT-ALL and AT-TABS need the side condition \( X \not\in \text{dom}(\Gamma) \), and various rules use the untyped normalization procedure introduced above.

7.3 Algorithmic Context Formation

\[
\begin{align*}
\emptyset & \vdash_A \text{ok} & (\text{AC-EMPTY}) \\
\Gamma, \ x: \text{ok} & \vdash_A A : \star & x \not\in \text{dom}(\Gamma) & (\text{AC-VAR}) \\
\Gamma & \vdash_A A : K & \text{nf}(K) \equiv \text{nf}(K') & X \not\in \text{dom}(\Gamma) & (\text{AC-TVAR})
\end{align*}
\]
7.4 Algorithmic Kind Formation

\[
\Gamma \vdash_A * \\
\Gamma, X \leq A : K_1 \vdash_A K_2 \quad \Gamma \vdash_A A : K_1' \\
\Gamma \vdash_A K_1 \quad \text{nfs}(K_1) \equiv \text{nfs}(K_1') \quad X \notin \text{dom}(\Gamma) \\
\Gamma \vdash_A \Pi X \leq A : K_1, K_2
\]

(\text{AK-I})

7.5 Algorithmic Type Formation

\[
\Gamma \vdash_A T_* : * \\
\Gamma_1, X \leq A : K, \Gamma_2 \vdash_A X : K \\
\Gamma \vdash_A A_1 : * \quad \Gamma \vdash_A A_2 : * \\
\Gamma \vdash_A A_1 \rightarrow A_2 : * \\
\Gamma \vdash_A K \quad \Gamma, X \leq A_1 : K \vdash_A A_2 : * \\
\Gamma \vdash_A A_1 : K' \quad \text{nfs}(K) \equiv \text{nfs}(K') \quad X \notin \text{dom}(\Gamma) \\
\Gamma \vdash_A \forall X \leq A_1 : K, A_2 : * \\
\Gamma \vdash_A K_1 \quad \Gamma, X \leq A_1 : K_1 \vdash_A A_2 : K_2 \\
\Gamma \vdash_A A_1 : K_1' \quad \text{nfs}(K_1) \equiv \text{nfs}(K_1') \quad X \notin \text{dom}(\Gamma) \\
\Gamma \vdash_A \lambda X \leq A_1 : K_1, A_2 : \Pi X \leq A_1 : K_1', K_2 \\
\Gamma \vdash_A A : \Pi X \leq B : K_1, K_2 \quad \Gamma \vdash_A C \leq B \\
\Gamma \vdash_A C : K_1' \quad \text{nfs}(K_1) \equiv \text{nfs}(K_1') \\
\Gamma \vdash_A A \: C : K_2[X \leftarrow C]
\]

(\text{AT-ALL})

(\text{AT-TABS})

(\text{AT-TAPP})

7.6 Equivalence of the Algorithm and \(F_{\leq}^0\)

In order to show the equivalence of the algorithm and the original system, we introduce an auxiliary notion of weak-head conversion that is particularly well-behaved in our setting under inversion, and we show some simple properties of this notion.

\textbf{Definition 7.6} Two types are \textit{weak-head convertible}, \(A \downarrow_{w} B\), or convert-
\(A \downarrow_n B\), if they are in the least relation such that:

\[
\begin{align*}
X & \downarrow_w X \\
T & \downarrow_w T \\
A_1 \rightarrow A_2 & \downarrow_w B_1 \rightarrow B_2 \quad \text{if } A_i \downarrow_n B_i \\
A C & \downarrow_w B C \quad \text{if } A \downarrow_w B, C \text{ normal and } A \text{ a variable or application} \\
\Lambda X \leq A_1 : K.A_2 & \downarrow_w \Lambda X \leq B_1 : K'.B_2 \quad \text{if } A_i \downarrow_n B_i \text{ and } K \downarrow_n K' \\
\forall X \leq A_1 : K.A_2 & \downarrow_w \forall X \leq B_1 : K'.B_2 \quad \text{if } A_i \downarrow_n B_i \text{ and } K \downarrow_n K' \\
A & \downarrow_n B \quad \text{if } A \rightarrow_w A', B \rightarrow_w B' \text{ and } A' \downarrow_w B'
\end{align*}
\]

with the obvious extension to kinds and contexts.

These untyped definitions are motivated by their inversion properties. For example, if we want to know under what circumstances \(\forall X \leq A_1 : K.A_2 \downarrow_w B\), then by inspection of the rules we see immediately that \(B \equiv \forall X \leq B_1 : K'.B_2\) and that \(A_i \downarrow_n B_i \) and \(K \downarrow_n K'\). This property does not follow so trivially for the equivalent notion of conversion formulated in Lemma 7.7 Case 3.

**Lemma 7.7**

1. If \(A \downarrow_w A'\) and \(HV(A)\) is undefined then \(HV(A')\) is undefined.
2. If \(X(A_1, \ldots, A_m) \downarrow_w B\) then \(B \equiv X(A_1, \ldots, A_m)\).
3. *If \(A \rightarrow_n C\) and \(B \rightarrow_n C\) then \(A \downarrow_n B\).*
   - *If \(A \rightarrow_w C\) and \(B \rightarrow_w C\) then \(A \downarrow_w B\).*

**Proof:** By induction on derivations, using simple inversion properties. \(\square\)

**Lemma 7.8**

1. If \(\Gamma \vdash_S \text{ok}\) then \(\Gamma \downarrow_n \Gamma\).
2. If \(\Gamma \vdash_S A : K\) then \(A \downarrow_n A\).

**Proposition 7.9**

- *If \(\Gamma \vdash_A A \leq B, \Gamma \downarrow_n \Gamma', A \downarrow_n A'\) and \(B \downarrow_n B'\) then \(\Gamma' \vdash_A A' \leq B'\).*
- *If \(\Gamma \vdash_A A \leq_w B, \Gamma \downarrow_n \Gamma', A \downarrow_w A'\) and \(B \downarrow_w B'\) then \(\Gamma' \vdash_A A' \leq_w B'\).*

**Proof:** By induction on derivations. We consider several cases.

**AS-INC** Then \(A \rightarrow_w C, B \rightarrow_w D\) and \(\Gamma \vdash_A C \leq_w D\). By inversion of \(A \downarrow_n A'\) we know \(A \rightarrow_w C'', A' \rightarrow_w C'\) and \(C'' \downarrow_w C'\), and since \(\rightarrow_w\) is deterministic we know \(C \equiv C''\), so \(C \downarrow_w C'\). Similarly we obtain \(B' \rightarrow_w D'\) with \(D \downarrow_w D'\), so by the induction hypothesis \(\Gamma' \vdash_A C' \leq_w D'\), and by AS-INC \(\Gamma' \vdash_A A' \leq B'\).

**AWS-TAPP** By Lemma 7.7 Case 2 we know that \(A' \equiv X(A_1, \ldots, A_m)\), and similarly for \(B'\). Also \(\Gamma'(X) \rightarrow_n B\) by assumption, so \(\Gamma' \vdash_A E \leq_w C'\).
PROPOSITION 7.10 (Correctness of the Algorithm)

(1) If $\Gamma \vdash_A \text{ok}$ then $\Gamma \vdash \text{ok}$.
(2) Given $\Gamma \vdash A K$ then $\Gamma \vdash A K$.
(3) Given $\Gamma \vdash \text{ok}$, if $\Gamma \vdash_A A : K$ then $\Gamma \vdash A : K$.
(4) Given $\Gamma \vdash A, B : K$. If $\Gamma \vdash_A A \leq_W B$ then $\Gamma \vdash A \leq B : K$.
(5) Given $\Gamma \vdash A, B : K$. If $\Gamma \vdash_A A \leq B$ then $\Gamma \vdash A \leq B : K$.

PROOF: By induction on derivations, using Soundness (Corollary 5.13), Completeness (Proposition 4.12), and Subject Reduction (Corollary 4.17). We show here the AT-TAPP case of item 3 to illustrate the proof technique.

We have that $\Gamma \vdash_A A : \Pi_X \leq_B K_1. K_2$, $\Gamma \vdash_A C \leq B$, $\Gamma \vdash_A C : K''_1$, and $\text{nf}(K_1) \equiv \text{nf}(K''_1)$. By the induction hypothesis 3, $\Gamma \vdash A : \Pi_X \leq_B K_1. K_2$ and $\Gamma \vdash C : K''_1$. In order to apply the induction hypothesis 5, we need to show that $\Gamma \vdash B : K_1$ and $\Gamma \vdash C : K_1$.

By Soundness, there exists $K'$ such that $\Gamma \vdash_S A : K'$ and $\Gamma \vdash_S \Pi_X \leq_B K_1. K_2 \Rightarrow K'$. Since kind normalization is deterministic, $K' \equiv \Pi_X \leq_B K'_1. K'_2$, $\Gamma \vdash_S B \Rightarrow K'_1$, and $\Gamma \vdash_S K_1 \Rightarrow K'_1$. By Lemma 7.4 Case 1, it follows that $\text{nf}(K_1) \equiv K'_1$. In the sequel, we shall use without mentioning that $\vdash$ is included in $\vdash$. By Completeness, $\Gamma \vdash B : K'_1$ and $\Gamma \vdash K_1 = \beta K'_1$. By K-Eq-Sym and T-Conv, $\Gamma \vdash B : K_1$.

From $\Gamma \vdash C : K''_1$, by Kind-Agreement, $\Gamma \vdash K''_1$, and from $\Gamma \vdash_S K_1 \Rightarrow K'_1$, by Completeness, $\Gamma \vdash K_1$. Then, by Lemma 7.5, $\Gamma \vdash K''_1 = \beta K_1$, and, by T-Conv, $\Gamma \vdash C : K_1$.

We can now apply the induction hypothesis 5 and obtain $\Gamma \vdash C \leq B : K_1$.

Finally, by T-TAPP, $\Gamma \vdash A C : K_2[X \leftarrow C]$.

PROPOSITION 7.11 (Completeness of the Algorithm)

(1) If $\Gamma \vdash_S \text{ok}$ then $\Gamma \vdash_A \text{ok}$.
(2) If $\Gamma \vdash_S K$ then $\Gamma \vdash_A K$.
(3) If $\Gamma \vdash_S A : K$ then $\Gamma \vdash_A A : K$ and $\text{nf}(K') \equiv K$.
(4) If $\Gamma \vdash_S A \leq_W B : K$ then $\Gamma \vdash_A A \leq_W B$.
(5) If $\Gamma \vdash_S A \leq B : K$ then $\Gamma \vdash_A A \leq B$.

PROOF: By induction on derivations. We show here a few cases.

SC-TVAR By Prefix (Lemma 4.6), there exists a subderivation of $\Gamma \vdash_S \text{ok}$.

Hence by the induction hypothesis 1, $\Gamma \vdash_A \text{ok}$. By the induction hypothesis 3, $\Gamma \vdash_A A : K''$ and $\text{nf}(K''') \equiv K'$. From $\Gamma \vdash_S K \Rightarrow K'$,
it follows that \( K \rightarrow_n K' \), by Lemma 7.4 Case 1. Hence, \( \text{nf}(K'') \equiv \text{nf}(K) \). By the induction hypothesis 2, \( \Gamma \vdash A \). Finally, by AC-TVAR, \( \Gamma, X \subseteq A : K \vdash A \).

**ST-BETA** By the induction hypothesis 3, \( \Gamma \vdash A : B : J \) and \( \text{nf}(J) \equiv \Pi X \subseteq A' : K'_1, K_2 \). By inversion, \( \Gamma \vdash S C \leq A : K'_1 \) can only be derived using SS-Inc. Hence, \( \Gamma \vdash S C : K'_1 \) with a proper subderivation, so we apply the induction hypothesis 3 and obtain \( \Gamma \vdash A C : J' \) and \( \text{nf}(J') \equiv K'_1 \). Moreover, by the induction hypothesis 5, \( \Gamma \vdash A C \leq A \).

By Adequacy on the first premise \( B \rightarrow_{\beta_2} \lambda X \subseteq A : K_1, D \), and by Subject Reduction \( \Gamma \vdash S \lambda X \subseteq A : K_1, D : \Pi X \subseteq A' : K'_1, K_2 \). Because the only rule to derive the latter is ST-TABS, we have that \( \Gamma \vdash S A \rightarrow_n A' : K'_1 \), and by Lemma 7.4 Case 2, \( A \rightarrow_n A' \).

Since \( \text{nf}(J) \equiv \Pi X \subseteq A' : K'_1, K_2 \), then \( J \equiv \Pi X \subseteq A'' : J_1, J_2 \) with \( \text{nf}(A'') \equiv A' \), \( \text{nf}(J_1) \equiv K'_1 \) and \( \text{nf}(J_2) \equiv K_2 \), by the definition of \( \rightarrow_n \). Hence, \( \Gamma \vdash A : J \equiv \Pi X \subseteq A'' : J_1, J_2 \), \( \Gamma \vdash A C : J' \) and \( \text{nf}(J') \equiv K'_1 \equiv \text{nf}(J_1) \).

To be able to apply AT-TAPP we need to prove that \( \Gamma \vdash A C \leq A'' \).

We know that \( \Gamma \vdash A C \leq A, A \rightarrow_n A', \) and \( A'' \rightarrow_n A' \), so \( A'' \downarrow_n A \), by Lemma 7.7 Case 3. By Lemma 7.8, \( C \downarrow_n C \). Then, by Proposition 7.9, \( \Gamma \vdash A C \leq A'' \).

By AT-TAPP, \( \Gamma \vdash A : BC : J_2[X \leftarrow C] \). We still need to prove that \( J_2[X \leftarrow C] \rightarrow_n K \). By Correctness of the Algorithm, \( \Gamma \vdash BC : J_2[X \leftarrow C] \), by Soundness, there exists \( K' \) such that \( \Gamma \vdash S J_2[X \leftarrow C] \rightarrow_n K' \), and, by Subject Reduction, \( \Gamma \vdash S K_2[X \leftarrow C] \rightarrow_n K' \), because \( J_2 \rightarrow_{\beta} K_2 \), by Lemma 7.1 Case 1. By Determinacy, \( K \equiv K' \). Finally, \( J_2[X \leftarrow C] \rightarrow_n K \), by Lemma 7.4 Case 1.

**SWS-TAPP** By Lemma 7.4 Case 2, the induction hypothesis 4, and AWS-TAPP.

The last two properties together with the Soundness of the semantics (Corollary 5.13) prove that the algorithm is sound and complete with respect to \( \mathcal{F}^{2}_{\leq} \).

**Proposition 7.12 (Equivalence of the Algorithm and \( \mathcal{F}^{2}_{\leq} \))**

(1) \( \Gamma \vdash A \) ok iff \( \Gamma \vdash \).
(2) \( \Gamma \vdash \) ok and \( \Gamma \vdash A : K \) iff \( \Gamma \vdash K \).
(3) \( \Gamma \vdash K, \Gamma \vdash A : K' \), and \( \text{nf}(K) \equiv \text{nf}(K') \) iff \( \Gamma \vdash A : K \).
(4) \( \Gamma \vdash A, B : K \) and \( \Gamma \vdash A \leq B \) iff \( \Gamma \vdash A \leq B : K \).

**Proof:**

(1) If \( \Gamma \vdash A \) ok then \( \Gamma \vdash \), by Correctness of the Algorithm (Proposition 7.10).

If \( \Gamma \vdash \) then by Soundness (Corollary 5.13) \( \Gamma \vdash S \) ok, and, by Completeness of the Algorithm (Proposition 7.11), \( \Gamma \vdash A \) ok.

55
(2) If $\Gamma \vdash \text{ok}$ and $\Gamma \vdash_{A} K$ then $\Gamma \vdash K$, by Correctness of the Algorithm.

If $\Gamma \vdash K$ then by Soundness, there exists a $K'$ such that $\Gamma \vdash_{S} K \Rightarrow_{n} K'$, and by Completeness (Proposition 4.12) $\Gamma \vdash_{A} K$.

(3) If $\Gamma \vdash K$, $\Gamma \vdash_{A} A : K'$, and $\text{nf}(K) \equiv \text{nf}(K')$ then by the Prefix Lemma $\Gamma \vdash \text{ok}$. By Correctness of the Algorithm $\Gamma \vdash : A : K'$, by KIND-AGREEMENT $\Gamma \vdash K'$, and by Lemma 7.5 $\Gamma \vdash K' =_{\beta} K$. Finally, by T-CONV, $\Gamma \vdash A : K$.

If $\Gamma \vdash A : K$ then by Soundness there exists a $K_1$ such that $\Gamma \vdash_{S} A : K_1$ and $\Gamma \vdash_{S} K \Rightarrow_{n} K_1$. By Lemma 7.4 Case 1 $\text{nf}(K) \equiv K_1$. By Completeness of the Algorithm, $\Gamma \vdash_{A} A : K'$ and $\text{nf}(K') \equiv K_1$. Hence $\text{nf}(K) \equiv \text{nf}(K')$, and $\Gamma \vdash K$ follows from $\Gamma \vdash A : K$ by KIND-AGREEMENT.

(4) If $\Gamma \vdash A, B : K$ and $\Gamma \vdash_{A} A \leq B$ then $\Gamma \vdash A \leq B : K$, by Correctness of the Algorithm.

If $\Gamma \vdash A \leq B : K$ then by Soundness there exists a $K''$ such that $\Gamma \vdash_{S} K \Rightarrow_{n} K''$ and $\Gamma \vdash_{S} A \leq B : K''$. By Completeness of the Algorithm, $\Gamma \vdash_{A} A \leq B$. By Completeness (Proposition 4.12), $\Gamma \vdash A, B : K''$ and $\Gamma \vdash K =_{\beta} K''$. Finally, by K-EQ-SYM and T-CONV, $\Gamma \vdash A, B : K$. □

By the equivalence, we can use the following sequence to check whether $\Gamma \vdash A \leq B : K$:

1. check that $\Gamma$ is a good context, $\Gamma \vdash_{A} \text{ok}$,
2. infer kinds $K'$ and $K''$ such that $\Gamma \vdash_{A} A : K'$ and $\Gamma \vdash_{A} B : K''$,
3. check that the given kind is well formed, $\Gamma \vdash_{A} K$,
4. check that $K, K', K''$ have the same normal form (which exist by Strong Normalization (Lemma 4.20)),
5. check that $\Gamma \vdash_{A} A \leq B$.

If any of the steps fails then the statement $\Gamma \vdash A \leq B : K$ is not derivable in $\mathcal{F}_{\leq},$ and otherwise it is.

Hence, the only significant result that remains to be proved for $\mathcal{F}_{\leq}$ is the decidability of type-checking and subtyping. These follow from the termination of the subtyping algorithm, as we see in the next section.

8 Decidability of Subtyping

In this section we prove that the algorithm from Section 7 constitutes a decision procedure for $\mathcal{F}_{\leq}$. As we have already shown that under suitable conditions the algorithmic rules compute the same relation as the rules of $\mathcal{F}_{\leq}$ (Proposition 7.12), only termination remains to be shown.

We divide our proof into four parts. We first prove that termination of algorithmic subtyping is preserved under normalization or expansion of types and
kinds in the context. We then divide out the main properties needed in the
proof of termination of the algorithm into a Key Lemma (Lemma 8.2). Next,
we show that algorithmic subtyping terminates on inputs typed in the typed
operational semantics (Theorem 8.3 and Corollary 8.4). This is where the
strength of the typed operational semantics is crucial. Finally, we prove that
the other judgements terminate under certain conditions (Proposition 8.5). We
demonstrate that when the algorithm is used these conditions are satisfied.

8.1 Preservation of Subtype-Checking Termination Under Context Conversion

To check if a given subtyping statement $\Gamma \vdash_A A \leq B$ holds, we use the
algorithmic subtyping rules to try to construct a derivation ending with $\Gamma \vdash_A A \leq B$. There are three possibilities for such a process:

- it may terminate with success, constructing such a derivation;
- it may terminate with failure, reaching a point where no rule is applicable:
  in other words, there is a statement that does not unify with the conclusion
  of any algorithmic rule, or
- the process may fail to terminate.

From this observation it is clear that we cannot prove termination just by
induction on derivations in the algorithmic system.

We instead define termination of algorithmic subtyping of a goal $\Gamma \vdash_A A \leq B$
or $\Gamma \vdash_A A \leq W B$ by saying that if, for every rule unifying with our goal checking
the premises of that rule terminates, then checking our goal terminates.
This is a well-foundedness property similar to our definition of strong normalization.
We can perform induction on derivations of this principle in the same
way that we can perform induction on derivations of strong normalization: we
call this principle Induction on Terminating Subtype Checking.

The following termination results will be proved using this principle.

**Lemma 8.1 (Preservation of Termination Under Context Conversion)**  
(1) If
$A \rightarrow_n B$ and $K \rightarrow_n K'$ then $\Gamma, X \leq A; K, \Delta \vdash_A C \leq D$ terminates if and
only if $\Gamma, X \leq B; K', \Delta \vdash_A C \leq D$ terminates.
(2) If $A \rightarrow_n B$ and $K \rightarrow_n K'$ then $\Gamma, X \leq A; K, \Delta \vdash_A C \leq_W D$ terminates if
and only if $\Gamma, X \leq B; K', \Delta \vdash_A C \leq_W D$ terminates.

**Proof:** By simultaneous induction on terminating subtype checking.

(1) Suppose $\Gamma, X \leq A; K, \Delta \vdash_A C \leq D$ terminates. The only rule matching
the conclusion is $\text{AS-INC}$ with premises $C \rightarrow_w C', D \rightarrow_w D'$, and
\[ \Gamma, X \leq A:K, \Delta \vdash A \quad C' \leq_W D'. \] By Lemma 7.3, \( C' \) and \( D' \) are unique. By the induction hypothesis, \( \Gamma, X \leq B:K', \Delta \vdash A \quad C \leq_D D' \) terminates. Hence, with AS-INC \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_D D \) terminates, and since there is no other rule matching our goal, this case is proved. Furthermore, AS-INC matches all conclusions, therefore there cannot be failure on unification.

The argument from right to left is identical.

(2) (a) Suppose \( \Gamma, X \leq A:K, \Delta \vdash_A C \leq_W D \) terminates. There are two cases to consider. Either \( \Gamma, X \leq A:K, \Delta \vdash_A C \leq_W D \) does not match the conclusion of any rule, in which case \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) does not match the conclusion of any rule either, and terminates with failure, or we have to consider every rule matching \( \Gamma, X \leq A:K, \Delta \vdash_A C \leq_W D \) as follows.

**AWS-Top** Since \( \text{HV}(C) \) is undefined, the only rule matching \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) is AWS-Top, so applying this rule terminates with success, and all other rules terminate with failure.

**AWS-Tapp** Let \( C = Y(A_1, \ldots A_m) \).

If \( Y \equiv X \), the premises are as follows: \((\Gamma, X \leq A:K, \Delta)(X) = A \Rightarrow_n B \) by assumption, \( B(A_1, \ldots A_m) \Rightarrow_w E \) and \( \Gamma, X \leq A:K, \Delta \vdash A \quad E \leq_W D \), where checking all three premises terminates. Observe that \((\Gamma, X \leq B:K', \Delta)(X) = B \Rightarrow_n B \) is an axiom in the definition of \( \Rightarrow_n \), and by the induction hypothesis on the third premise \( \Gamma, X \leq B:K', \Delta \vdash_A E \leq_W D \) terminates. Hence, since the only rule matching \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) is AWS-TAPP, and all other rules fail, checking \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) always terminates.

If \( Y \neq X \), the premises are as follows: \((\Gamma, X \leq A:K, \Delta)(Y) = B', B'(A_1, \ldots A_m) \Rightarrow_w E \) and \( \Gamma, X \leq A:K, \Delta \vdash A \quad E \leq_W D \), where checking all three premises terminates. Notice that \((\Gamma, X \leq A:K, \Delta)(Y) = (\Gamma, X \leq B:K', \Delta)(Y)\), and that, by the induction hypothesis on the last premise, \( \Gamma, X \leq B:K', \Delta \vdash_A E \leq_W D \) terminates. Hence, since the only rule matching \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) is AWS-TAPP, and all other rules fail, checking \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) always terminates.

**AWS-Refl** By AWS-Refl and matching failure.

**AWS-Arrow** By the induction hypothesis, AWS-Arrow, and matching failure.

**AWS-All** By the induction hypothesis, AWS-All, and matching failure.

**AWS-Tabs** By the induction hypothesis, AWS-Tabs, and matching failure.

(b) The argument from right to left is almost identical. The only difference appears in the AWS-TAPP case in the subcase when \( Y \equiv X \), which we consider here.

Suppose \( \Gamma, X \leq B:K', \Delta \vdash_A C \leq_W D \) terminates and \( C \equiv X(A_1, \ldots A_m) \).
The premises are as follows: \((\Gamma, X \leq B; K', \Delta)(X) \vdash_n B, B(A_1, \ldots, A_m) \vdash_w E\) and \(\Gamma, X \leq B; K', \Delta \vdash_A E \leq_W D\), where checking all three premises terminates. Observe that \((\Gamma, X \leq A; K, \Delta)(X) = A \vdash_n B\) is an assumption, and by the induction hypothesis on the third premise \(\Gamma, X \leq A; K, \Delta \vdash_A E \leq_W D\) terminates. Hence, since the only rule matching \(\Gamma, X \leq A; K, \Delta \vdash_A C \leq_W D\) is AWS-TAPP, and all other rules terminate with failure, checking \(\Gamma, X \leq A; K, \Delta \vdash_A C \leq_W D\) always terminates.  

\[\Box\]

### 8.2 The Key Lemma

The proof of Theorem 8.3, Termination of Subtyping, involves technical details about decidability of subtyping with respect to weak-head normal forms which are factored out in the following Key Lemma. The Key Lemma gives sufficient conditions for the termination of subtyping, and the proof of Termination of Subtyping shows that these conditions are met. The proof of the Key Lemma is more involved than many of the proofs by induction we are used to doing in type systems; we have an induction on the derivation of a type reduction statement, and for each type reduction rule we consider all possible algorithmic subtyping rules that may match that case. The measure that decreases is the derivation tree of a type reduction statement.

**LEMMA 8.2 (Key Lemma)**

1. If \(\Gamma \vdash_S G \rightarrow_w B \rightarrow_n B' : K', \Gamma \vdash_S \Gamma(X) \rightarrow_n C : K, \Gamma \vdash_S C(A_1, \ldots, A_m) \rightarrow_w E : K\) and \(\Gamma \vdash_A E \leq_W B\) terminates then \(\Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W B\) terminates.
2. If \(\Gamma \vdash_S B \rightarrow_w C \rightarrow_n D : K\) then \(\Gamma \vdash_A C \leq_W X(A_1, \ldots, A_m)\) terminates.
3. Suppose that \(\Gamma \vdash_S B \rightarrow_n B' : \ast\), and that \(B \equiv B_1 \rightarrow B_2\) implies that \(\Gamma \vdash_A B_1 \leq A_1\) terminates and \(\Gamma \vdash_A B_2 \leq A_2\) terminates. Then \(\Gamma \vdash_A A_1 \rightarrow A_2 \leq_W B\) terminates.
4. Suppose that \(\Gamma \vdash_S B \rightarrow_n B' : \ast\), and that \(B \equiv B_1 \rightarrow B_2\) implies that \(\Gamma \vdash_A A_1 \leq B_1\) terminates and \(\Gamma \vdash_A B_2 \leq A_2\) terminates. Then \(\Gamma \vdash_A B \leq_W A_1 \rightarrow A_2\) terminates.
5. Suppose that \(\Gamma \vdash_S K_1 \rightarrow_n K'_1, \Gamma \vdash_S A_1 \rightarrow_n E_1 : K'_1, \Gamma \vdash_S D \rightarrow_n E : \ast\), and that if \(D \equiv \forall X \leq D_1; K.D_2, \Gamma \vdash_S K \rightarrow_n K'_1\) and \(\Gamma \vdash_S D_1 \rightarrow_n E_1 : K'_1\) then \(\Gamma, X \leq A_1; K \vdash_A A_2 \leq D_2\) terminates. Then \(\Gamma \vdash_A \forall X \leq A_1; K.A_2 \leq_W D\) terminates.
6. Suppose that \(\Gamma \vdash_S K_1 \rightarrow_n K'_1, \Gamma \vdash_S A_1 \rightarrow_n E_1 : K'_1, \Gamma \vdash_S D \rightarrow_n E : \ast\), and that if \(D \equiv \forall X \leq D_1; K.D_2, \Gamma \vdash_S K \rightarrow_n K'_1\) and \(\Gamma \vdash_S D_1 \rightarrow_n E_1 : K'_1\) then \(\Gamma, X \leq D_1; K \vdash_A A_2 \leq A_2\) terminates. Then \(\Gamma \vdash_A D \leq_W \forall X \leq A_1; K.A_2\) terminates.
7. Suppose that \(\Gamma \vdash_S K_1 \rightarrow_n K'_1, \Gamma \vdash_S A_1 \rightarrow_n E_1 : K'_1, \Gamma \vdash_S D \rightarrow_n E : \ast\), and that if \(D \equiv \forall X \leq D_1; K.D_2, \Gamma \vdash_S K \rightarrow_n K'_1\) and \(\Gamma \vdash_S D_1 \rightarrow_n E_1 : K'_1\) then \(\Gamma, X \leq D_1; K \vdash_A A_2 \leq A_2\) terminates. Then \(\Gamma \vdash_A D \leq_W \forall X \leq A_1; K.A_2\) terminates.
$E : \Pi X \leq E_1 : K_1', K_2$, and that if $D \equiv \Lambda X \leq D_1 : K.D_2$, $\Gamma \vdash_S K \rightarrow_n K_1'$ and $\Gamma \vdash_S D_1 \rightarrow_n E_1 : K_1'$ then $\Gamma$, $X \leq A_1 : K_1 \vdash_A A_2 \leq D_2 : K_2$. Then $\Gamma \vdash_A \Lambda X \leq A_1 : K_1.A_2 \leq_W D$ terminates.

(8) Suppose that $\Gamma \vdash_S K_1 \rightarrow_n K_1'$, $\Gamma \vdash_S A_1 \rightarrow_n E_1 : K_1'$, $\Gamma \vdash_S D \rightarrow_n E : \Pi X \leq E_1 : K_1'.K_2$, and that if $D \equiv \Lambda X \leq D_1 : K.D_2$, $\Gamma \vdash_S K \rightarrow_n K_1'$ and $\Gamma \vdash_S D_1 \rightarrow_n E_1 : K_1'$ then $\Gamma$, $X \leq D_1 : K \vdash_A D_2 \leq A_2 : K_2$. Then $\Gamma \vdash_A D \leq_W \Lambda X \leq A_1 : K_1.A_2$ terminates.

(9) If $\Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K$ then $\Gamma \vdash_A T_* \leq_W B$ terminates.

(10) If $\Gamma \vdash_S A \rightarrow_w B \rightarrow_n C : K$ then $\Gamma \vdash_A B \leq_W T_*$ terminates.

**Proof:**

(1) This statement follows by induction on the derivation of $\Gamma \vdash_S G \rightarrow_w B \rightarrow_n B' : K$. For each possible last rule in this derivation we show that $\Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W B$ terminates by considering every algorithmic subtyping rule.

This case is the only case in this proof where for every type reduction rule, there is the possibility that the given subtyping statement might terminate with success. In other words, there is always a matching subtyping rule (AWS-TAPP), or the result follows by induction, as is the case for ST-BETA. We consider each of the rules:

**ST-TOP** Then $B \equiv T_*$. We now show that for every algorithmic subtyping last rule, $\Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W T_*$ terminates. Except for the rule AWS-TAPP, all rules terminate with failure. We now consider the case AWS-TAPP. We have to prove that:

(a) $\Gamma(X) \rightarrow_n C$ terminates, meaning that there exists such a $C$.

(b) $C(A_1, \ldots, A_m) \rightarrow_w E$ terminates, meaning that there exists such an $E$.

(c) $\Gamma \vdash_A E \leq_W T_*$ terminates.

By assumption and Lemmas 7.4 and 7.3, we have that 1a and 1b hold, and by assumption we have that 1c holds. The cases for ST-ARROW and ST-ALL are similar.

**ST-TVAR** Then $B \equiv Y$. We consider the last possible rules to derive $\Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W Y$ in order to show that it terminates. The case AWS-TAPP is as for ST-TOP, since the rule is independent of the form of $B$. The case AWS-REFL terminates with success if $X \equiv Y$ and $m = 0$, and otherwise it terminates with failure, and all other algorithmic subtyping rules terminate with failure.

The case for ST-TAPP is similar.

**ST-BETA** This case follows by the induction hypothesis, because there is a subderivation of $\Gamma \vdash_S G \rightarrow_w B \rightarrow_n B' : K$ deriving $\Gamma \vdash_S F \rightarrow_w B \rightarrow_n B' : K$.

(2) This case follows by induction on the derivation that $\Gamma \vdash_S B \rightarrow_w C \rightarrow_n D : K$. In most cases, there is no subtyping rule matching the statement we are trying to prove, and subtyping checking terminates with failure. For
example, consider the case ST-ARROW: the subtyping statement we need
to check is of the form \( \Gamma \vdash_A B_1 \rightarrow B_2 \leq_W X(A_1, \ldots, A_m) \), for which there is no possible last rule.

The only three cases that do not terminate with immediate failure
for lack of a matching subtyping rule are when \( \Gamma \vdash_S B \rightarrow_w C \overset{\text{w}}{\rightarrow} D : K \) follows from ST-BETA, ST-TVAR, or ST-TAPP. The case ST-BETA
follows by the induction hypothesis, because there is a subderivation with
expression \( \Gamma \vdash_S B' \rightarrow_w C \overset{\text{w}}{\rightarrow} D : K \).

Consider the case ST-TAPP. Then \( C \equiv Y(B_1, \ldots, B_n) \). The only two
rules that may match \( \Gamma \vdash_A Y(B_1, \ldots, B_n) \leq_W X(A_1, \ldots, A_m) \) are AWS-
REFL, which immediately terminates, and AWS-TAPP. To show that the
latter terminates we need to show that:
(a) \( \Gamma(Y) \rightarrow_n F \) terminates, meaning that there exists such an \( F \).
(b) \( F(B_1, \ldots, B_m) \rightarrow_w E \) terminates, meaning that there exists such an \( E \).
(c) \( \Gamma \vdash_A E \leq_W X(A_1, \ldots, A_m) \) terminates.

We know that \( \Gamma \vdash_S \Gamma(Y) \rightarrow_n F : K' \) and \( \Gamma \vdash_S F(B_1, \ldots, B_n) \rightarrow_w E : K \) are premises of ST-TAPP, using Lemmas 7.4 and 7.3 to justify the
choice of \( F \) and \( E \). Furthermore, \( \Gamma(Y) \rightarrow_n F \) and \( F(B_1, \ldots, B_n) \rightarrow_w E \) are finite, and \( \Gamma \vdash_A E \leq_W X(A_1, \ldots, A_m) \) terminates by the induction hypothesis, because the derivation of \( \Gamma \vdash_S F(B_1, \ldots, B_n) \rightarrow_w E : K \) is a
subderivation of the derivation of \( \Gamma \vdash_S B \rightarrow_w Y(B_1, \ldots, B_n) \rightarrow_n D : K \),
which is a premise of ST-TAPP.

The case ST-TVAR follows the same argument as that of the case ST-
TAPP.

(3) The proof is by induction on the derivation of \( \Gamma \vdash_S B \overset{\text{w}}{\rightarrow} B' : \ast \). In most
cases there is no subtyping rule matching the statement \( \Gamma \vdash_A A_1 \rightarrow A_2 \leq_W B \) we are trying to derive, and subtyping terminates with failure. The only
two cases where \( \Gamma \vdash_A A_1 \rightarrow A_2 \leq_W B \) does not terminate immediately
with failure are when \( \Gamma \vdash_S B \overset{\text{w}}{\rightarrow} B' : \ast \) has been obtained by ST-
BETA, ST-ARROW or ST-TOP. The case ST-BETA follows by the induction hypothesis, because there is a subderivation with conclusion \( \Gamma \vdash_S H \overset{\text{w}}{\rightarrow} B \overset{\text{w}}{\rightarrow} B' : \ast \).

In the case "ST-ARROW," \( B \equiv B_1 \rightarrow B_2 \), and \( \Gamma \vdash_S (B_1 \rightarrow B_2) \rightarrow_w (B_1 \rightarrow B_2) \overset{\text{w}}{\rightarrow} \overset{\text{w}}{\rightarrow} (E_1 \rightarrow E_2) : \ast \). The only rule to derive \( \Gamma \vdash_A A_1 \rightarrow A_2 \leq_W B_1 \rightarrow B_2 \) is AWS-ARROW. Then \( \Gamma \vdash_A B_1 \leq A_1 \) and \( \Gamma \vdash_A A_2 \leq B_2 \) terminate by assumption.

In the case ST-TOP, \( B \equiv T \), and the only rule matching \( \Gamma \vdash_A A_1 \rightarrow A_2 \leq_W T \) is AWS-TOP, which immediately terminates with success.

(4) This case is symmetric to Case 3. The only difference is that the cases
where \( \Gamma \vdash_A B \leq_W A_1 \rightarrow A_2 \) does not terminate immediately with failure
are when \( \Gamma \vdash_S B \overset{\text{w}}{\rightarrow} B' : \ast \) has been obtained by ST-BETA, ST-ARROW,
ST-TVAR, or ST-TAPP.

ST-BETA follows by the induction hypothesis and ST-ARROW follows

61
as in Case 3. ST-TVAR and ST-TAPP are similar to Case 2, but with no possibility of applying AWS-Refl.

(5) The proof is by induction on the derivation of $\Gamma \vdash S \ D \warrow E : \star$. The only cases that do not terminate with immediate failure for lack of a matching subtyping rule are when $\Gamma \vdash S \ D \warrow E : \star$ follows from ST-TOP, ST-BETA, or ST-ALL.

In the case ST-TOP, where $D \equiv T_\star$, all algorithmic subtyping rules terminate immediately with failure except for AWS-TOP, which immediately terminates with success.

In the case ST-BETA, the result follows by the induction hypothesis, because there is a subderivation of $\Gamma \vdash S \ D \warrow E : \star$ deriving $\Gamma \vdash S \ A' \warrow D \warrow E : \star$.

For the case ST-ALL, let $D \equiv \forall X : D_1 ; K : D_2$. The only subtyping rule that does not immediately terminate with failure is AWS-ALL. To prove that this rule terminates, we need to prove that:
(a) $\text{nf}(A_1) \equiv \text{nf}(D_1)$ terminates,
(b) $\text{nf}(K_1) \equiv \text{nf}(K)$ terminates, and
(c) $X \leq A_1 ; K_1 \vdash A_2 \leq D_2$ terminates.

For 5a, by assumption $\Gamma \vdash S \ A_1 \warrow E_1 : K_1'$, and by inversion of ST-ALL, $\Gamma \vdash S \ D_1 \warrow E_1' : K'$. Then 5a terminates by Lemma 7.4. If the normal forms $E_1$ and $E_1'$ are different then our goal terminates with failure. Otherwise $E_1 \equiv E_1'$, and we proceed to check 5b.

Again, by assumption $\Gamma \vdash S \ K_1 \warrow K_1'$, and by inversion $\Gamma \vdash S \ K \warrow K'$, Therefore 5b terminates by Lemma 7.4. If $K_1'$ and $K'$ are not equal our goal terminates with failure. Otherwise $K_1' \equiv K'$, and we proceed to check 5c.

Now, by assumption $\Gamma, X \leq A_1 ; K_1 \vdash A_2 \leq D_2$ terminates.

(6) This case is similar to the previous one, but the cases where $\Gamma \vdash A \ D \leq \forall X \leq A_1 ; K_1 . A_2$ does not terminate immediately with failure are when $\Gamma \vdash S \ D \warrow E : \star$ has been obtained by ST-BETA, ST-ALL, ST-TVAR, or ST-TAPP. The reasoning for ST-ALL is as for the previous case, and ST-BETA, ST-TVAR and ST-TAPP follow as for Case 2, but with no possibility of applying AWS-Refl.

(7) Similar to Case 5.

(8) This case follows by induction on the derivation of $\Gamma \vdash S \ D \warrow E : \forall X \leq A_1' ; K_1 . K_2$.

The only cases that do not terminate with immediate failure for lack of a matching subtyping rule are when $\Gamma \vdash S \ D \warrow E : \forall X \leq A_1' ; K_1 . K_2$ follows from ST-BETA, ST-TVAR, ST-TAPP, and ST-TABS. The case ST-BETA follows by the induction hypothesis. The cases ST-TAPP and ST-TVAR are as in Case 2, but with no possibility of applying AWS-Refl.

Consider now the case ST-TABS. Let $D \equiv \Delta X \leq D_1 ; K ; D_2$. Now, the only matching subtyping rule is AWS-TABS. Therefore we have to show that:
(a) $\text{nf}(A_1) \equiv \text{nf}(D_1)$ terminates,
(b) \( \text{nf}(K_1) \equiv \text{nf}(K) \) terminates, and

(c) \( \Gamma, X \leq D_1 : K \vdash_A D_2 \leq A_2 \) terminates.

Items 8a and 8b are as before in Case 5, and by assumption \( \Gamma, X \leq D_1 : K \vdash_A D_2 \leq A_2 \) terminates.

(9) The proof is by induction on the derivation of \( \Gamma \vdash_S A \Rightarrow_{w} B \Rightarrow_{n} C : K \).

In the cases ST-TVAR, ST-TAPP, ST-ARROW, ST-ALL, and ST-TABS, there is no algorithmic rule matching \( \Gamma \vdash_A T_* \leq_W B \), therefore all these cases terminate with failure.

For ST-TOP, the rule AWS-TOP terminates with success and all other rules terminate with failure. The case ST-BETA follows by the induction hypothesis.

(10) We prove that \( \Gamma \vdash_A B \leq_W T_* \) terminates by induction on the derivation of \( \Gamma \vdash_S A \Rightarrow_{w} B \Rightarrow_{n} C : K \).

ST-ARROW Let \( B \equiv B_1 \Rightarrow B_2 \). The only rule to prove \( \Gamma \vdash_A B_1 \Rightarrow B_2 \leq_W T_* \) is AWS-TOP, which immediately terminates with success, and the other rules terminate with failure.

The cases for ST-ALL and ST-TABS are similar.

ST-BETA The result follows by the induction hypothesis.

ST-TOP Then \( B \equiv T_* \). The rule AWS-TOP terminates with success, while all other rules terminate with failure.

ST-TAPP Let \( B \equiv X(A_1, \ldots, A_m) \). Then \( \Gamma \vdash_S \Gamma(X) \Rightarrow_{n} D : K \) and \( \Gamma \vdash_S D(A_1, \ldots, A_m) \Rightarrow_{w} E : * \) are premises of ST-TAPP. By the induction hypothesis \( \Gamma \vdash_A E \leq_W T_* \) terminates, and by Case 1, \( \Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W T_* \) terminates.

The case for ST-TVAR is similar.

\[ \square \]

8.3 Termination

We now show the principal new result of this section, Termination of Subtyping, and develop its consequences.

As we show below, the premises for bound replacement of ST-TAPP, in particular the premise of the form \( \Gamma \vdash_S D(A_1, \ldots, A_m, F) : K \), are essential to the following theorem, showing that the subtyping algorithmic rules define a total relation on well-kindred types. The premise allows us to know by the induction hypothesis that the algorithm is terminating for the replacement of the variable by its bound. This is the information needed in the Key Lemma (Lemma 8.2 Case 2) to show termination of the algorithm for the variable case.

**Theorem 8.3 (Termination of Subtyping)** If \( \Gamma \vdash_S A \Rightarrow_{w} C \Rightarrow_{n} E : K \) and \( \Gamma \vdash_S B \Rightarrow_{w} D \Rightarrow_{n} F : K \) then:

63
(1) \( \Gamma \vdash_A C \leq_W D \) and \( \Gamma \vdash_A D \leq_W C \) terminate.

(2) \( \Gamma \vdash_A A \leq B \) and \( \Gamma \vdash_A B \leq A \) terminate.

**PROOF:** The proof is by induction on the derivation of \( \Gamma \vdash_S A \rightarrow_w C \rightarrow_n E : K \) (the inner inductions on \( \Gamma \vdash_S B \rightarrow_w D \rightarrow_n F : K \) occur in Lemma 8.2).

(1) (a) Termination of \( \Gamma \vdash_A C \leq_W D \).

**ST-TOP** \( C \equiv T_\ast \) follows by Lemma 8.2 Case 9.

**ST-TVAR** Consider \( C \equiv x. \). \( \Gamma \vdash_S \Gamma(x) \rightarrow_w C' \rightarrow_n B' : K' \) and \( \Gamma \vdash_S B' \rightarrow_n B'' : K'' \) are subderivations. By Lemma 4.4 and Adequacy \( B' \equiv B'' \), so by the induction hypothesis (Case 1a) \( \Gamma \vdash_A B' \leq_W D \) terminates. Taking \( m = 0 \), Lemma 8.2 Case 1 implies that \( \Gamma \vdash_A X \leq_W D \) terminates.

**ST-TAPP** Consider \( C \equiv X(A_1, \ldots, A_m) \). We have a subderivation \( \Gamma \vdash_S D(A_1, \ldots, A_m) \rightarrow_w E' : K \), so by the induction hypothesis (Case 1a), \( \Gamma \vdash_A E' \leq_W D \) terminates. We also have a subderivation \( \Gamma \vdash_S \Gamma(X) \rightarrow_n D' : K' \).

Hence by Lemma 8.2 Case 1, \( \Gamma \vdash_A X(A_1, \ldots, A_m) \leq_W D \) terminates.

**ST-ARROW** \( C \equiv C_1 \rightarrow C_2 \). We have subderivations of \( \Gamma \vdash_S C_1 \rightarrow_w A_1 \rightarrow_n E_1 : \ast \) and \( \Gamma \vdash_S C_2 \rightarrow_w A_2 \rightarrow_n E_2 : \ast \). Suppose \( D \equiv D_1 \rightarrow D_2 \). By inversion and the induction hypothesis (Case 2), \( \Gamma \vdash_A D_1 \leq C_1 \) terminates and \( \Gamma \vdash_A C_2 \leq D_2 \) terminates.

Hence, by Lemma 8.2 Case 3, \( \Gamma \vdash_A C_1 \rightarrow C_2 \leq_W D \) terminates.

**ST-ALL** \( C \equiv \forall X \leq C_1 : K_1, C_2 \). We have subderivations of \( \Gamma \vdash_S K_1 \rightarrow_n K_1' \), \( \Gamma \vdash_S C_1 \rightarrow_n E_1 : K_1' \) and \( \Gamma, X \leq C_1 : K_1 \vdash_S C_2 \rightarrow_w D_2 \rightarrow_n F_2 : \ast \).

Suppose \( D \equiv \forall X \leq D_1 : K, D_2 \). Then by inversion, Determinacy (Lemma 4.11), Context Conversion (Lemma 4.23), Lemma 4.22 Case 1, and SCN-TVAR in Definition 4.21, we have that \( \Gamma, X \leq C_1 : K_1 \vdash_S D_2 : \ast \), so by the induction hypothesis (Case 2) we have that \( \Gamma, X \leq C_1 : K_1 \vdash_A C_2 \leq D_2 \) terminates.

Hence by Lemma 8.2 Case 5, \( \Gamma \vdash_A \forall X \leq C_1 : K_1, C_2 \leq_W D \) terminates.

**ST-TABS** Similar to the case for ST-ALL, using Lemma 8.2 Case 7.

**ST-BETA** By the induction hypothesis on the premise of the form \( \Gamma \vdash_S A' \rightarrow_w C \rightarrow_n E : K \).

(b) Termination of \( \Gamma \vdash_A D \leq_W C \).

**ST-TOP** \( C \equiv T_\ast \) follows by Lemma 8.2 Case 10.

**ST-TVAR** \( C \equiv X. \) By Lemma 8.2 Case 2 with \( m = 0 \).

**ST-TAPP** \( C \equiv X(A_1, \ldots, A_m) \). By Lemma 8.2 Case 2.

**ST-ARROW** Similar to the ST-ARROW case in the proof of termination of \( \Gamma \vdash_A C \leq_W D \), using Lemma 8.2 Case 4.

**ST-ALL** Similar to the next case for ST-TABS, using Lemma 8.2
Case 6.

ST-TABS \( C \equiv \Lambda X \leq C_1 : K_1, C_2 \). We have subderivations of \( \Gamma \vdash S K_1 \rightarrow_n K'_1 \), \( \Gamma \vdash S C_1 \rightarrow_n E_1 : K'_1 \), and \( \Gamma, X \leq C_1 : K_1 \vdash S C_2 : K_2 \).

Suppose \( D \equiv \Lambda X \leq D_1 : K, D_2 \), \( \Gamma \vdash S K \rightarrow_n K'_1 \) and \( \Gamma \vdash S D_1 \rightarrow_n E_1 : K'_1 \). By inversion, Determinacy (Lemma 4.11), Context Conversion (Lemma 4.23), Lemma 4.22 Case 1, and SCN-TVAR in Definition 4.21, we have \( \Gamma, X \leq C_1 : K_1 \vdash S D_2 : K_2 \), so by the induction hypothesis \( \Gamma, X \leq C_1 : K_1 \vdash_A D_2 \leq C_2 \) terminates. Finally, \( C_1 \rightarrow_n E_1, D_1 \rightarrow_n E_1, K_1 \rightarrow_n K'_1 \) and \( K \rightarrow_n K'_1 \) by Lemma 7.4, so applying Lemma 8.1 in both directions \( \Gamma, X \leq D_1 : K \vdash_A D_2 \leq C_2 \) terminates.

The result follows using Lemma 8.2 Case 8.

ST-BETA By the induction hypothesis on the premise of the form
\[ \Gamma \vdash_A A' \rightarrow_w C \rightarrow_n E : K. \]
(2) (a) Termination of \( \Gamma \vdash_A A \leq B \). We now show that \( \Gamma \vdash_A A \leq B \) terminates. The only rule matching this is AS-INC, with premises \( A \rightarrow_w C', B \rightarrow_w D' \), and \( \Gamma \vdash_A C' \leq W D' \). By Lemma 7.4 Case 2 and Lemma 7.3, we have that \( C \equiv C' \) and \( D \equiv D' \). Because \( A \) and \( B \) are well-kind in the semantics, \( A \rightarrow_w C \) and \( B \rightarrow_w D \) are finite reductions. Finally, we prove that \( \Gamma \vdash_C C \leq W D \) terminates as in Case 1.

(b) Termination of \( \Gamma \vdash_A B \leq A \). Similar to the above case for \( \Gamma \vdash_A A \leq B \). \( \square \)

**Corollary 8.4 (Termination of Subtyping) If** \( \Gamma \vdash_S A \rightarrow_w C \rightarrow_n E : K \) and \( \Gamma \vdash_S B \rightarrow_w D \rightarrow_n F : K \) then \( \Gamma \vdash_A C \leq_D D \) and \( \Gamma \vdash_A A \leq B \) terminate.

The next step is to establish the decidability of the other algorithmic judgements.

**Proposition 8.5 (Termination)** If \( \Gamma \vdash ok \) then \( \Gamma \vdash_A A : K \) and \( \Gamma \vdash_A K \) terminate. Furthermore, \( \Gamma \vdash_A ok \) terminates.

**Proof:** Termination of \( \Gamma \vdash_A A : K \) and \( \Gamma \vdash_A K \) follow by simultaneous induction on the structure of the subject, using Soundness, Equivalence (Proposition 7.12) and Termination of Subtyping. Termination of \( \Gamma \vdash_A ok \) follows by induction on the structure of \( \Gamma \) using the previous result. \( \square \)

### 8.4 Decidability of Kinding

The following sequence can be used to check \( \Gamma \vdash A \leq B : K \). First, check that \( \Gamma \) is a good context, \( \Gamma \vdash_A ok \), which by Proposition 8.5 terminates. Then, infer kinds \( K' \) and \( K'' \) such that \( \Gamma \vdash_A A : K' \) and \( \Gamma \vdash_A B : K'' \).
both of which terminate by Proposition 8.5. Check that the given kind is well formed, \( \Gamma \vdash_A K \), which terminates by Proposition 8.5, and check that \( K, K', K'' \) have the same normal form \( K''' \) (which exists by Strong Normalization). Hence, \( \Gamma \vdash S A : K''' \) and \( \Gamma \vdash S B : K''' \) by Equivalence (Proposition 7.12) and Soundness. Finally, check that \( \Gamma \vdash_A A \leq B \), which terminates by Theorem 8.3.

If any of the steps fails then the statement \( \Gamma \vdash A \leq B : K \) is not derivable in \( \mathcal{F}_{\leq} \), and otherwise it is.

Hence we have proved that kinding and subtyping for \( \mathcal{F}_{\leq} \) are decidable.

**Theorem 8.6 (Decidability of Kind Formation, Kinding, and Subtyping)**

1. \( \Gamma \vdash K \) is decidable.
2. \( \Gamma \vdash A : K \) is decidable.
3. \( \Gamma \vdash A \leq B : K \) is decidable.

We can then apply the usual technique for showing decidability of typing based on these results.

## 9 Related and Future Work

Bruce [8] uses bounded operator abstraction, but does not develop the metatheory. Compagnoni [23] mentions the open problem of studying the metatheory for bounded operator abstraction.

Most type systems with subtyping do not have the circularity between type formation and subtyping mentioned in the introduction: for example, \( \mathcal{F}_{\leq} \) [13, 14, 15, 17, 46], \( \mathcal{F}_< \) [25], and the systems in Abadi and Cardelli’s book on objects [2] all separate the two judgements. One system that does have the circularity is \( \lambda P_{\leq} \), a system for subtyping with dependent types studied by Aspinall and Compagnoni [4]. There, the authors avoid the interdependency by finding a particular order in which to prove results.

We used the foundations established in this paper to prove anti-symmetry of \( \mathcal{F}_{\leq} \) [24], the property that if \( \Gamma \vdash A \leq B : K \) and \( \Gamma \vdash B \leq A : K \) then \( \Gamma \vdash A =_\beta B \). That proof is the first proof of anti-symmetry for a type theory with higher-order subtyping. The property leads to an easier treatment of the metatheory, by allowing equality to be defined in terms of subtyping, and also permits a simpler and more efficient implementation of subtyping and equality by defining them jointly. The property was difficult to prove because it only holds for well-formed judgements, similar to the Church–Rosser property for \( \beta \eta \)-reduction, and because it requires a logical relation incorporating information about the bounds of variables, as formulated in our rule ST-TAPP.
As we mentioned in Section 1, the model construction is based on well-established ideas in dependent type theory. Streicher [51] gives a partial interpretation function to define the categorical semantics of the calculus of constructions, a technique which is now widely used. Coquand and Gallier [27] introduce Kripke-style models to build typed proofs of strong normalization for systems with dependent types. Typed operational semantics has been used to develop the metatheory of UTT, a sophisticated type theory with inductive types, impredicative propositions and type universes [32, 33]. Coquand [26] interprets judgemental equality as a logical relation to show properties of Martin-Löf type theory with $\beta\eta$-equality, similar to our interpretation of the subtyping relation.

It seems to be possible to use the technique developed by the first author for higher-order subtyping [23] instead of the development with typed operational semantics for the particular system $\mathcal{F}_{\leq}$ that we study here. We believe that substitution can first be proved simultaneously for the kinding and subtyping judgements for the original system (without the structural rules). This can then be used to prove the subject reduction property for the original system, which in turn is used to establish basic properties of the normal system appropriately formulated for $\mathcal{F}_{\leq}$. However, this approach does not enjoy the advantages of typed operational semantics mentioned in Section 1.2. In particular, the admissibility of the structural rules in Section 2.3 needs to be proved by induction on derivations for each individual rule, the overall proof is delicate and based on a particular order for the results, and the benefits of typed operational semantics for studying properties of reduction such as subject reduction and strong normalization are lost.

The syntax for types and contexts could be simplified by removing kind annotations. It seems that this would lead to a more efficient algorithm, because checks between the given and inferred kinds, involving normalization of both kinds, would disappear. Moreover, the system would be closer to an implementable system, placing less burden on the user to supply kind information. The model construction in Section 5 could be adapted to a system with less information along the lines of Streicher’s book [51].

There are several directions for future work. The proof here should easily extend to a system with $\beta\eta$-equality, the equality for which typed operational semantics was originally developed. We also believe that the model construction can be extended to cope with $\Gamma$-reduction, replacing variables $X$ by their bounds $A$ if $X \leq A : K$ is in $\Gamma$, which cannot be done directly in the semantics because of an interdependency of transitivity elimination and context replacement. Finally, we have not included recursive types or objects, but Abadi and Cardelli [2] have demonstrated that these do not present difficulties at the level of types, and our proof should extend without any problems.
10 Conclusions

In this paper we have studied $\mathcal{F}_{\leq}$, the first treatment of the metatheory for a system of higher-order subtyping with bounded operator abstraction. We have used techniques for constructing models for dependent type theory to solve problems associated with the weak dependency introduced by bounds in kinds, and we have modeled the subtyping relation directly rather than using a syntactic encoding. We have also used the new tool of typed operational semantics to give simpler proofs for meta-theoretic properties such as substitution, kind agreement, and subject reduction and Church–Rosser for type reduction. Finally, we have shown the equivalence with the algorithmic presentation of the system. Because the techniques introduced are adapted from other contexts and do not involve encodings of syntax, we believe that they are generally applicable.

Acknowledgments

We thank Luca Cardelli for providing encouragement and motivation for this work, and Paul Jackson and Martin Steffen for their comments on drafts of the paper. We are also grateful to the anonymous referees for their comments. The first author is funded by a European Community TMR grant, and the second author is funded by a research fellowship from EPSRC.

References

[27] T. Coquand and J. Gallier. A proof of strong normalization for the theory
of constructions using a Kripke-like interpretation. In Workshop on Logical
Development 97, Lecture Notes in Computer Science. Springer-Verlag, Apr.
1997.
[29] P.-L. Curien and G. Ghelli. Coherence of subsumption: Minimum typing and
type-checking in $F_\leq$. Mathematical Structures in Computer Science, 2:55–91,
[33] H. Goguen. The metatheory of UTT. In Types for Proofs and Programs, volume
996 of Lecture Notes in Computer Science, pages 60–82, Baastad, Sweden, June
[34] H. Goguen. Typed operational semantics. In Proceedings of the International
Conference on Typed Lambda Calculi and Applications, volume 902 of Lecture
[38] R. Harper and M. Lillibridge. A type-theoretic approach to higher-order mod-
ules with sharing. In Proceedings of the Twenty-first ACM Symposium on
Principles of Programming Languages, pages 123–137, Portland, OR, January
1994.
aspects of coercive subtyping. In E. Giménez and C. Paulin-Mohring, editors,
Proceedings of TYPES’96, volume 1512 of Lecture Notes in Computer Science,
1996.
[40] M. P. Jones. A system of constructor classes: overloading and implicit higher-
[41] X. Leroy. Applicative functors and fully transparent higher-order modules. In
[42] D. MacQueen. Using dependent types to express modular structure. In Pro-
cedings of the Thirteenth ACM Symposium on the Principles of Programming
Languages, 1986.
J. F. Groote, editors, Proceedings of the International Conference on Typed
Lambda Calculi and Applications, pages 289–305. Springer-Verlag, LNCS 664,