Polarized Higher-Order Subtyping

Der Technischen Fakultät der
Universität Erlangen-Nürnberg

zur Erlangung des Grades

D O K T O R – I N G E N I E U R

vorgelegt von

Martin Steffen

Erlangen – 1997

Summary

The calculus of higher order subtyping, known as $F^{\omega}_{\leq}$, a higher-order polymorphic λ-calculus with subtyping, is expressive enough to serve as core calculus for typed object-oriented languages. The versions considered in the literature usually support only pointwise subtyping of type operators, where two types $S\ U$ and $T\ U$ are in subtype relation, if S and T are. In the widely cited, unpublished note [Car90], Cardelli presents $F^{\omega}_{\leq}$ in a more general form going beyond pointwise subtyping of type applications in distinguishing between monotone and antimonotone operators. Thus, for instance, $T\ U_1$ is a subtype of $T\ U_2$, if $U_1 \leq U_2$ and T is a monotone operator.

My thesis extends $F^{\omega}_{\leq}$ by polarized application, it explores its proof theory, establishing decidability of polarized $F^{\omega}_{\leq}$. The inclusion of polarized application rules leads to an interdependence of the subtyping and the kinding system. This contrasts with pure $F^{\omega}_{\leq}$, where subtyping depends on kinding but not vice versa. To retain decidability of the system, the equal-bounds subtyping rule for all-types is rephrased in the polarized setting as a mutual-subtype requirement of the upper bounds.

Contents

Chapter 1

Introduction

The introductory chapter collects and reviews relevant research in the field of typed (functional) calculi for object-oriented languages, at the same time introducing some object-oriented jargon. The material is well-known and included for reference. I also give an overview of the rest of the thesis and provide references for background literature.

1.1 Typed programming

Types are a natural concept in programming. The type of a program characterizes its allowed use and strong typing means, that no run-time type error can occur by illegitimate use of a program, such as feeding an argument into a procedure it was not designed for, or invoking a method on an object it doesn't support. A language is called type-safe, if such errors are caught by type checking. For statically typed languages this check is carried out at compile-time, before actually running the program.

The type system constitutes a formal system, dividing all syntactically correct programs into well-typed and ill-typed ones. This protection against accidental errors increases the chances of catching errors early, contributing to more reliable software. Moreover, assigning a type to each program at compile-time frees the run-time system from costly checks during the execution of the program. In documenting the intended usage of a program, types can lead to more readable code. Especially for large, structured programs, types specify the interfaces of a module or an object and type checking amounts to a consistency check of the modules' interfaces. Moreover compilers use type information to generate more efficient code.

Not everything that might go wrong at run-time is statically checkable (because it is undecidable whether a program has some undesired property or it is simply too inefficient). Since a static, safe type system has to be conservative in this respect, it will reject programs that would otherwise run well, and the weaker the type system, the more programs it will reject. Thus the design of expressive, yet safe and decidable type systems is a major theme in the evolution of programming languages and an important contribution of theoretical computer science. The development of provably safe, expressive type systems is especially challenging for object-oriented languages, which are known for a large array of different, advanced features, not all too well understood.

1.2 Object-oriented programming

Objects and classes

Even if there is not too much agreement about what the array of features of an object-oriented language should consist of, one thing is for sure: programs are arranged in *objects*. An object contains an internal *state* together with procedures, called *methods*, to manipulate it. The only way to access the state of an object is via its methods, building the object's interface; thus objects form an encapsulation barrier around their internals. The support of clean interface mechanisms and abstraction is crucial for structuring large programs by libraries. (See [Coo91] for a discussion of the differences between abstract data types and objects.)

A salient feature of object-oriented languages is *late-binding* of methods, also known as dynamic binding, dynamic look-up, message passing, or run-time dispatch of methods. Late-binding of a method means that the code executed for a method is not selected statically at compile-time, when the object is constructed, but at run-time, when the method is invoked by message passing. This is a fundamental difference between function application and message-passing. Since methods may be defined referring to other methods of the same object (via *self* in Smalltalk or *this* in C^{++}, for instance), a re-implementation of the referenced methods, called method *override*, affects the referring methods of the same object, as well. It is plausible that this dynamic feature is complicated to capture in static type systems.

The actual program is structured into objects, but whether the program development centers around objects is a separate question. In class-based languages, such as Simula [BDMN79], Smalltalk [GR83], C^{++} [Str86] [ES90], Eiffel [Mey92], Java [AG96] [LY96], to name a few, *classes* are used for incremental program construction.[1] They serve as blueprint for objects, containing the description of their implementation, i.e., some initial values for the internal data and the code for the methods. Classes can be used in two ways. First, to create new objects sharing the implementation common to all objects belonging to the class, its *instances*. Second, to incrementally define new classes by *inheritance*, where parts of the old superclass may be used, i.e., inherited, old methods replaced or "overridden", and new methods added. Thus, inheritance is a tool for constructing programs, supporting reuse of code of superclasses by their subclasses through inheritance, and structuring large programs or libraries into an inheritance hierarchy.

Polymorphism

Like all high-level languages deserving the name, object-oriented languages support polymorphism in one form or the other. A polymorphic program, as opposed to a monomorphic one, accepts inputs of different types, adding flexibility and expressiveness to a language (cf. Cardelli and Wegner [CW85] for a discussion of different forms of polymorphism.)

Parametric polymorphism Parametric polymorphism allows the formulation of typed programs working uniformly on input of different types by providing the program with the type as *parameter*.

[1]Class-based languages constitute the mainstream of object-oriented languages. Alternatively, object-based languages, e.g. Self [US91], do without classes, performing inheritance and the creation of new objects directly on objects.

A standard illustrating example is the function swapping the components of a pair of elements of a given type. There is nothing specific to the concrete type involved and the function works uniformly for all possible types. So instead of writing a specific function of type $(T \times T) \rightarrow (T \times T)$ for each type T for which the function is needed in the rest of the program, it is clearly preferable to uniformly define a function for all types, attaching to it the universally quantified type $\forall A.(A \times A) \rightarrow (A \times A)$, where A is a type variable.

This type characterizes functions which take as first argument the type which they are supposed to work on. Polymorphic functions, taking types as parameters by type abstraction and using type application to instantiate them is an elegant and powerful tool for code reuse, leading to clear and concise programs. Furthermore it can help the compiler to avoid code duplication. For object-oriented languages, parametric polymorphism can be used to avoid type-casts resulting in more efficient code.

This form of parametric polymorphism is called explicit, in contrast to widely used polymorphic type systems of modern languages (e.g., in ML, Haskell, Miranda et. al.) where type *inference* algorithms free the programmer from explicitly providing the type as parameter to a function. These so-called Hindley-Milner type systems [Hin69] [Mil78] [DM82] (see also Cardelli [Car87]) or systems with "let-polymorphism" are weaker in that they assume the types of polymorphic functions to be implicitly quantified in prenex position, only. The advantage of these type systems is that, unlike for systems with unrestricted higher-order polymorphism [Boe85] [Pfe93a] [Wel94], type inference is decidable.

Subtype polymorphism Subtyping is based on a simple idea: types are ordered and a program of a smaller type can be used at places, where programs of a larger type are expected. Consider, as a not specifically object-oriented example, integer numbers and reals. Even if the type system should prevent using the real number at places where an integer is required, there is nothing wrong from the programmer's point of view, using conversely an integer instead of a real number, since after all he has learned, an integer is also a real number. The set of reals thus *subsumes* the set of integers and the type *Int* is said to be a subtype of *Real*, written $Int \leq Real$. Of course, from the compiler's point of view, integers and reals are probably represented differently and some measures have to be taken internally, to prevent for example the direct addition of the different representations, converting the representation of the integer number into a common floating point representation.[2]

[2]More generally, using programs non-uniformly on arguments is called *overloading*. In the example, the arithmetic operation $+$, adding two numbers, whether they are reals or integers, is said to be overloaded. Since the way the sum is computed for a pair of integers differs from the addition of two reals, overloading is also called "ad-hoc"-polymorphism, to contrast it with the uniform definitions

If subtyping were all about using at times integers instead of reals, it would not be too exciting. But besides atomic or base types such as the type of integers or of reals, a language usually possesses also compound types. If there is any types system at all, it is likely to contain functional types of the form $T_1 \rightarrow T_2$, describing all functions with input of type T_1 and output of type T_2. Considering the subtypes of the arrow type $T_1 \rightarrow T_2$ and guided by the intuition that values of such a smaller type are expected to be substituted safely for values (here functions) of the larger type $T_1 \rightarrow T_2$, the subtypes have to contain functions, as well. More specifically, the functions inhabiting the subtype must be able to accept *at least* all values of type T_1 as input, while yielding results ranging *at most* over T_2. This is captured in the following inference rule:

$$\frac{T_1 \leq S_1 \qquad S_2 \leq T_2}{S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2}$$

Thus arrow-types are said to behave contravariantly on the argument type, inverting the direction of the subtype relation, and covariantly on the result type.

The idea that the subtype relation is determined by the structure of the types is called *structural* subtyping [Car88a] [Car88b] [Rey85] [Sta88].[3] The subtype relation, generally a preorder on all types, is inductively defined by a set of inference rules similar to the one for arrow types above, which constitute the *subtype system.*

The essence of subtyping, namely that elements or programs of a smaller type also carry all larger types and thus may safely be substituted for those of the larger types, is expressed in the rule of *subsumption*, which connects the typing system with the subtyping system:

$$\frac{t \in T' \qquad T' \leq T}{t \in T}$$

Subtype polymorphism, also called inclusion polymorphism, is characteristic for object-oriented languages. For example, an object supporting more methods can safely replace an object with less methods without incurring a run-time error such as "message-not-understood"; thus the types of both objects are in subtype relation.

Subtype-bounded polymorphism An important contribution for typed object-oriented languages is the combination of parametric or universal polymorphism and subtype polymorphism. In the seminal paper [CW85], Cardelli and Wegner proposed

of parametric polymorphism.

[3]This is in contrast to *declarative* subtyping, where it is up to the user to declare certain types to be in subtype relation. That some typed object-oriented languages, e.g., C^{++} or Eiffel, do not distinguish between classes and types adds to the confusion.

"Bounded Fun", a second-order polymorphic λ-calculus with bounded subtyping and records as a functional model for object-oriented languages. Let's take the standard toy example for illustration: point objects with an internal state denoting its coordinates and two methods for manipulating it. Thus assuming *Point* as the type of all such point objects supporting *setx* and *getx* as methods to read and write to its x-coordinate allows us type the invocation of the *getx*-method, for instance, with $\forall A{\leq}Point.A \to Int$, where A ranges over all subtypes of *Point*. The structural definition of the subtype relation guarantees that all elements for all subtypes of *Point* share some common structural properties, in the example of point-objects assuring that all objects of a subtype of *Point* support at least a *getx*- and a *setx*-method of the appropriate type.

To account for the more complex situation where, unlike in the *getx*-example of above, a method changes the state of an object, for example the *setx*-method of the intended type $\forall A{\leq}Point.A \to Int \to A$, the second-order calculus with subtype-bounded polymorphism has been extended or changed in various ways, for instance by recursion, F-bounded polymorphism, match-bounded polymorphism, or higher-order polymorphism. Meanwhile, there are numerous languages supporting subtype-bounded polymorphism; among others Pizza, Rapide, Trellis/OWL [SCB$^+$86], Eiffel, polyTOIL ... On the other hand, the lack of parametric, subtype-bounded polymorphism is one major drawback in the design of the object-oriented language Java. Consequently, different proposals have been made to include parametric polymorphism, for example in [AFM97], or by Odersky and Wadler's Pizza-language [OW97b] [OW97a], both based of F-bounded polymorphism. Other work in this direction is done in [MBL97] [BLM96].

Inheritance As pointed out earlier, inheritance is a mechanism for the incremental construction of objects, reusing already defined code. On the other hand subtyping is about the external use of instantiated objects, not their implementation. It is commonly accepted that both concepts are different and should thus be separated. This contrasts with some languages identifying classes with types and hence inheritance with subtyping, e.g. C^{++} or Eiffel, where inheritance is the only way to produce subtypes. See [Sny86] [BHJ$^+$87] [CHC90] [LP91] and many others for a discussion of subtyping vs. inheritance.

Objects as records An often used intuition about objects regards them as records of their methods [CW85], where method invocation corresponds to record selection. In first approximation, also the intuition of subtyping of object types fits nicely with the subtyping rules for records: two record types are in subtype relation, if the smaller one contains all the fields of the larger, and possibly more, and the types of the common fields are related covariantly by subtyping (the fields l_i of the record are assumed to be distinct and their order does not play a role):

$$\frac{S_i \;\leq\; T_i \quad \text{for all } 1 \leq i \leq n}{\{l_1{:}S_1, \ldots, l_n{:}S_n, l_{n+1}{:}S_{n+1}, \ldots, l_{n+m}{:}S_{n+m}\} \;\leq\; \{l_1{:}T_1, \ldots, l_n{:}T_n\}}$$

Nearly all typed calculi for object-oriented languages are loosely related to this analogy. This simplistic view, though, has to be refined to account for the interplay of subtyping, inheritance with late-binding of methods and overriding, and encapsulation.

If especially in an object-based setting, the records or objects are the direct target of manipulations beyond method invocation, i.e. record selection, the analogy of subtyping between record types and object types breaks down. Record extension, adding new fields to a record, requires to abandon one half of the above subtyping rule for records, namely that a record type with more fields is a subtype of one with less, called width-subtyping; otherwise, by subsumption, one would be induced to extend an object by a method it already possesses, causing a run-time error. To deal with method addition or record extension, record calculi have been devised to include information about which methods are *not* already present in an object, and hence can be safely added.

On the other hand, method override, replacing the code for one method means to give up the second half of the above subtyping rule, namely that the types of the common fields are in subtype relation, called depth-subtyping, and require the types of the common fields of two records to coincide.

To deal with these problems, a number of record calculi have been proposed and investigated (especially wrt. type inference), allowing record update, record extension, or record concatenation [Wan87a] [Wan88] [Wan87b] [Car88a] [Wan91] [CM91] [Rém89] [Rém92] [HP91] [Car92] [Wan94] [Aba94] [JM93] [Hen94] [GJ96] [Zwa96] ... As a method can call methods — including itself — of the same object, recursive records have been studied to model this self-referencing. In [Coo87] [Coo89], classes are represented as functions, called object generators, and objects are recursively defined records as the fixed point of the generator. Other work on recursive records includes [CP89] [Red88] [KR94] [Bru92] [CHC90] [CCHM89] [Mit90b]. For an elaborate language design using recursive records see [Bru94].

Overloading Instead of belonging to an object, a method can be viewed (and implemented) as an entity on its own where depending on the type of the object it is invoked on different code is chosen. By placing the methods outside the objects, there is nothing specific to any object and the code chosen when executing the method may depend on more than one object. This is known as multiple-dispatch, as for example in CLOS [KG89] [DG87] or Cecil [Cha93] [Cha92]. The messages are modelled as functions with the receiving objects as parameters; thus messages are viewed as overloaded functions. An extension of simply typed λ-calculus for late-binding, subtyping, and overloaded functions called $\lambda\&$ is presented in [Cas93] [CGL95]. This can be applied to model multiple dispatch or multimethods. The system $F^{\&}_{\leq}$ [Cas95] is the extension to the second-order case. A meta-language λ_{object}, based on $\lambda\&$, is investigated in [Cas93]. See Castagna [Cas97] for a comprehensive account of modelling object-oriented languages based on multiple dispatch.

Matching Matching, written $S <\# T$, is a relation on types, similar to subtyping, but more general.[4] Matching, in particular, does not enjoy the subsumption property, which is the essence of subtyping. This relation has been proposed by Bruce and others as a generalization of the subtype relation. It allows the definition of classes where the type of allowed implementations ranges not over all subtypes but over all types which match the type of the current implementation. This is more general, as it allows to construct subclasses by inheritance, whose instances' types are not necessarily subtypes of the instances of the superclass [Bru94] [Bru96b] [BFP97] [AC96a]. Bounded polymorphism where the parameter is bounded by the matching relation instead of an upper bound as supertype is called *match-bounded* polymorphism or bounded matching. Matching can be encoded by full higher-order subtyping [AC96a]. Sample language designs using matching are Toil [BSvG93] and polyTOIL [BSvG95], statically typed languages with subtyping together with matching and imperative features. In the recent language Loom [BFP97] [Bru96b], subtyping is dropped altogether and replaced by matching. Further languages supporting match-bounded polymorphism are Theta [LCD+94] [DGLM95] and School [RIR93].

F-bounded polymorphism F-bounded subtype polymorphism, which was proposed in [CCHM89] [CHC90], can be seen as a specific form of higher-order subtyping. The upper bound S in the polymorphic type $\forall A{\leq}S.T$ may contain free occurrences of the type variable A, which makes it act like a function from types to types, and polymorphic function of this type accepts as type argument each type U with $U \leq [U/A]T$. See also [BCGŠ91] [KLMM94]. Languages supporting F-bounded polymorphism are Sather [OL92], Pizza, and a couple of others.

Object calculi

The mentioned functional calculi have in common, that they treat objects as a derived concept. Alternatively, a couple of formalisms have been proposed with objects as primitive notion. The most advanced array of such calculi, ranging from untyped object calculi and first-order calculi to higher-order ones with recursion, can be found

[4]Matching here should not be confused with pattern matching in high-level languages, such as ML.

in Abadi and Cardelli's book [AC96b]. Starting from an untyped object calculus ς-calculus [AC94], based on the notion of method update and late-binding[5], but without method extension the formalism is extended by first- order and higher-order type system and subtyping, all in an object-based setting. The object calculus is extended in [Liq97].

Other calculi are concerned with concurrent object-oriented programming. One example is the language Pict [PT97b][PT97c] based on the π-calculus [MPW92], a process algebra for mobile computation. Also in the setting of process calculi, typing is an important issue [PS96]. Whereas in a sequential, functional setting types impose a discipline on programs to avoid illegitimate function application, the way process calculi interact with their environment is more complex, namely by communication and synchronization. Here, static type systems are employed to enforce appropriate usage of channels for communication. In Pict, for example, a sophisticated version of $F^\omega_\leq$ is implemented, supporting recursive types, partial type inference, and pattern matching. Other concurrent object-oriented calculi include [Nie92] [DF96] [HT91] [HT92] [VT93]. Cardelli's object-based language Obliq [Car95], ABCL [Yon90], based on the actor model, and America's language Pool [Ame89]. Other process algebra based languages include [Vas94] [Nie92] [KY94] [Jon93].

1.3 $F^\omega_\leq$: the calculus of higher-order subtyping

The study of typed functional calculi, i.e. typed λ-calculi, flexible and expressive enough to provide a theoretical framework for object-oriented languages has received considerable attention in the literature over the past ten years. The object of study in this thesis is $F^\omega_\leq$ (called "F-omega-sub"), one prominent representative of λ-calculi for object-oriented programming. It is a higher-order, typed λ-calculus with subtyping and can be understood as the combination of its simpler fragments: the pure polymorphic λ-calculus or system F, the ω-order calculus F^ω, and the polymorphic λ-calculus with subtyping ($F_\leq$).

As starting point serves Girard and Reynold's *Système F* [Gir71] [Gir72] [Rey74] or the second order polymorphic λ-calculus. This system extends Church's simply typed λ-calculus [Chu40] by parametric polymorphism, adding type abstraction $\lambda A.t$ and type application $t\ T$, on the level of terms (where A is a type variable and T is a type). Adding type *operators*, i.e., functions from types to types, yields F^ω [Gir72], a system with higher-order polymorphism. On the other hand, extending the polymorphic λ-calculus with subtyping and bounded quantification yields $F_\leq$; Cardelli and Wegner's language Bounded Fun [CW85] is one variant of this system.

[5]The ς of the ς-calculus replaces λ as variable binder to remind that late binding of methods is different from static binding of functions.

The metatheoretical properties $F_{\leq}$ have been extensively studied in the literature (cf. [CMMŠ94] [Ghe91] [CG92] [BCGŠ91] [Pie94] et. al.). The integration of both subtyping and type operators leads to $F^{\omega}_{\leq}$ [Car90] [Mit90b], the calculus of higher-order subtyping. It is sufficiently expressive to model class inheritance with late binding, encapsulation, and subtype polymorphism in a uniform framework, and has thus been proposed as underlying core calculus for class-based typed object-oriented languages [PT94] [HP95b] in the style of Smalltalk. Objects are represented as elements of existential type, explicitly hiding the state and the implementation of the methods. By adding type operators it is possible, to represent not only the type of objects, but also the interfaces or signatures of objects as functions from types to types, namely taking the representation type and yielding the interface.

Implementation issues and extensions of $F_{\leq}$, such as partial type inference for second order types, de Bruijn indices, and a syntax extension mechanism, are discussed in [Car93]. Similarly for higher-order subtyping implemented by explicit coercions in [Cra97]. The KML-compiler [Cra96a] [Cra96b] is based on the explicitly typed λ^{K}-calculus, an extension of $F^{\omega}_{\leq}$ containing records, recursive types, singleton and power kinds. Pierce and Turner's [PT94] model can be represented in FCP [Jon97], a calculus with first-class polymorphism, type inference, and abstract data-types.

See [PT94] [HP95b] for a more detailed account of $F^{\omega}_{\leq}$'s object model. See also the surveys [FM96] and [BCP96] for a comparison of different object-models and encodings. A refinement of $F^{\omega}_{\leq}$'s object model to deal with friendly functions using partially abstract types (cf. [MP88]) is presented [PT93]. Extensions of the type systems to include recursive types can be found in [AC93] [MPS86] [CC91].

Semantical *models* based on partial equivalence relations (PER's) have proven useful for denotational semantics of subtyping and polymorphism. An important paper about semantics of subtyping based on partial equivalence relations is Bruce and Longo [BL90]. A PER-model for $F^{\omega}_{\leq}$ in can be found in [BM92] [AP90], likewise a model for $F_{\leq}$ with positive subtyping in [HP95a]. A coercion-based model is elaborated in [BCGŠ91]. Other material about model theory in this context can be found in [Pho90] [Ama91] [MV96] [FMRS92].

Variants

The subtyping calculi $F_{\leq}$, $F^{\omega}_{\leq}$, and related ones, come in different variants. An important distinctive feature is the subtyping rule for universal quantification. A simple version of the All-rule was proposed in [CW85] for the language "Kernel Fun", requiring that the upper bound of to All-types under comparison have to be identical:

$$\frac{\Gamma, A{\leq}U \;\vdash\; S_2 \;\leq\; T_2}{\Gamma \;\vdash\; \mathit{All}(A{\leq}U)S_2 \;\leq\; \mathit{All}(A{\leq}U)T_2} \qquad \text{(S-All-Kernel)}$$

A stronger and equally sensible formulation, here called S-All-Full, allows the upper bounds of the two types to vary contravariantly:

$$\frac{\Gamma, A{\leq}T_1 \;\vdash\; S_2 \;\leq\; T_2 \qquad \Gamma \;\vdash\; T_1 \;\leq\; S_1}{\Gamma \;\vdash\; \mathit{All}(A{\leq}S_1)S_2 \;\leq\; \mathit{All}(A{\leq}T_1)T_2} \qquad \text{(S-All-Full)}$$

Ghelli [Ghe95] showed that a previously proposed subtype "algorithm" can diverge in the presence of the full subtyping rule; Pierce [Pie94] additionally proved that this rule renders the subtyping relation for $F_{\leq}$ actually undecidable. Other investigations about the implications of this rule can be found in Ghelli [Ghe93]. In Curien and Ghelli [CG92], the full version of the quantifier rule is used in a study of $F_{\leq}$'s meta-theory based on coercions as explicit representation for the derivations of subtyping statements. By proof-rewriting techniques, they obtain a sound and complete semi-decision procedure for $F_{\leq}$. Similarly in [BCGŠ91], mapping $F_{\leq}$ to the polymorphic λ-calculus F with record types. The weaker rule S-All-Kernel leads to systems with much nicer properties than the more complex formulation and has been used in [Com95b] [PS97]. A further comparison of the two different rules in the presence of bounded existential types can be found in [GP97]. See also [Ghe90].

In this thesis, we propose yet another variant of the All-subtyping rule. On the one hand we strife to retain as much of the well-behavior of the kernel variant of $F^{\omega}_{\leq}$. On the other hand, the presence of polarized subtyping will lead us to relax the equal-bounds requirement of S-All-Kernel to a mutual-subtype requirement (denoted by $\gtreqless$):

$$\frac{\Gamma \;\vdash\; S_1 \;\gtreqless\; T_1 \;\in\; K_1 \quad \Gamma, A{\leq}S_1{:}K_1 \;\vdash\; S_2 \;\leq\; T_2 \;\in\; K}{\Gamma \;\vdash\; \mathit{All}(A{\leq}S_1{:}K_1)S_2 \;\leq\; \mathit{All}(A{\leq}T_1{:}K_1)T_2 \;\in\; \star} \quad \text{(S-All-Pol)}$$

In presence of polarized subtyping, this rule is strictly more general than the equal-bound subtyping rule, since, unlike for pointwise subtyping, the mutual subtyping relation between types will not imply β-equivalence. Based on the proof of decidability of polarized higher-order subtyping in Part II, we believe that this rule is the adequate generalization in the presence of polarized applications.

Extensions

In Hofmann and Pierce [HP95a], a "positive" variant of Kernel $F_{\leq}$ is proposed with the aim to capture the semantics of updating. Commonly, the subtype relation $S \leq T$ can be interpreted by a coercion function of type $S \to T$, converting elements of the smaller type S to elements of type T. Hofmann and Pierce deviate from this "orthodox" view, stipulating that $S \leq T$ requires not only the existence of a standard extraction function, coercing values of type S into values of type T, but also an *update*-function of type $S \to T \to S$, which, given an element of types S overwrites the "T-part" of it by the value of its second argument of type T. The introduction of a constant $put[S, T]$ on term level for each pair S and T to that effect leads to a stronger, more restrictive interpretation of the subtyping relation. For example, no subtyping is allowed between existential types, i.e., no object subtyping. Also, the subtyping rule for arrow types is restricted in that the argument types on both sides have to coincide. The resulting object-model with positive subtyping extends and simplifies the one from [PT94], for example resulting in a simpler representation of inheritance, as the subtype relation already comes equipped with update functions which otherwise needs explicitly to be programmed. The stronger interpretation also allows to reason about object-oriented programs in a structured way, as it allows to transfer properties of the methods from superclasses to subclasses. This model of positive subtyping is encoded in [HNSS98] in the type-theoretic proof-assistant Lego. Exploiting the correspondence between propositions and types or the Curry-Howard isomorphism [CF58] [How80] [GLT89], classes are extended by a proof-component, thus including proof methods into the encapsulated objects, where in much the same way as with ordinary, computational method, the actual proofs are hidden and only the properties are accessible via the methods interface.

Intersection types A couple of extensions generalize $F_{\leq}$ or $F^{\omega}_{\leq}$ to include *intersection* types. The resulting calculi are called $F_{\wedge}$ ("F-meet") [Pie97b] [Ma92] for the second-order case and $F^{\omega}_{\wedge}$ ("F-omega-meet") [Com95a] including type operators. The intuition of the intersection of S and T (written $S \wedge T$ for binary intersection of S and T) is that the values of intersection types carry both types. Using intersection types in the upper bound of polymorphic expressions for inheritance allows to express that the interface of the sub-class inherits from more than one super-class. This model of *multiple inheritance* via intersection types is developed in [CP96]. The system $F^{\omega}_{\wedge}$ is a proper generalization of $F^{\omega}_{\leq}$, where the maximal type *Top* can be understood as the empty intersection. For the treatment of intersection types in combination with higher-order subtyping the reader is referred to Compagnoni's thesis [Com95b] or [Com97a] [Com97b]. A language-design based on intersection types is Reynold's Forsythe [Rey96].

Dependent types $F^{\omega}_{\leq}$ is an impredicative type theory, where terms depend on terms and types, and types take types as parameters (in which case they are called type operators). Systems where types can depend on terms, are said to have dependent types. This allows to express functions, uniformly parametric for a range of terms. As a very simple example take the type of arrays of a fixed length n, whose type depends on n, a natural number. The strength of systems with dependent types goes far beyond the simple example of the type of arrays and can be used to specify properties of programs. Thus, these calculi form the basis of many type-theoretic proof-checkers such as Lego [leg97], Coq [CCF$^+$95], Isabelle [Pau93], Nuprl [C$^+$86] et. al. For a systematic classification of the different λ-calculi by the different combinations of dependence between types and terms in the so-called λ-cube, consult Barendregt's handbook article [Bar92].)

Recently, interest in calculi with dependent types extended by subtyping has started. Aspinall and Compagnoni [AC96c] investigate $\lambda P_{\leq}$, a first order calculus with dependent types (known as λP or $\lambda\Pi$) extended by subtyping, establishing decidability of the system. Chen [Che97a] studies a different formulation of the same system, which he calls $\lambda\Pi_{\leq}$. The inclusion of dependent types leads to a cyclic dependence of the typing, the subtyping, and the kinding statements, posing similar proof-theoretic problems as the study of polarized applications pursued in this thesis. A more detailed discussion and a comparison with the results of this work is left for the conclusions.

1.4 Background material

I assume some acquaintance with typed functional and object-oriented calculi. The reader is encouraged to consult the following works of a broader perspective.

Comprehensive material about typed functional programming languages, i.e., typed λ-calculi, their operational and denotational semantics, can be found in the books of Mitchell [Mit96] and Gunter [Gun92]. While both books focus on functional languages, they contain also some sections about subtype polymorphism. Mitchell especially discusses the functional record-model of object-oriented languages and bounded subtype polymorphism, including F-bounded and higher-order polymorphism. Similarly, but with more emphasis on type theory in its own right than on programming language design in Thompson's book [Tho91]. The latter also discusses type systems featuring dependent types, which find their application in proof checkers. Two handbook articles about type systems in programming languages are Mitchell [Mit90a] and more recently Cardelli [Car97]. For a discussion of the use of types in programming, with reference to many existing languages and also with respect to implementation issues, see Cardelli [Car91].

Whereas there are a couple of accessible references for the foundations of functional languages, less textbook material is available for the semantical issues of object-oriented languages. One comprehensive treatment and a standard reference is Abadi and Cardelli's book [AC96b], advocating the standpoint that objects are the fundamental notion for the investigation of object-oriented languages, and not functions. Thus the study presents different typed and untyped object-calculi with method-update, late-binding, and subtyping of varying expressiveness. More complex features such as classes and inheritance are explained via encoding in the basic calculi. Castagna's recent book [Cas97] similarly contains the theoretical treatment of object-oriented languages with multiple dispatch as in CLOS, and the mathematical formalisms here are λ-calculi with overloading. Type systems for class-based object-oriented languages with emphasis on type-inference algorithms are investigated in Palsberg and Schwarzbach's book [PS94].

Different approaches to the controversial issue of binary methods are discussed by several researchers in the field in [BCC$^+$96]. In [BCP96], different known object encodings are compared; the vehicle of comparison is the calculus $F^{\omega}_{\leq}$ with recursive types. In the survey paper [Bru96b], Bruce reviews a couple of typed calculi for object-oriented languages and argues in favor of match-bounded instead of subtype-bounded polymorphism. He illustrates his point by a couple of examples and referring to many existing object-oriented languages and current experimental language designs. Many influential, original papers about functional calculi for object-oriented programming are reprinted in [GM94].

1.5 Results and contribution of the thesis

The calculus of higher-order subtyping, known as $F^{\omega}_{\leq}$, a polymorphic λ-calculus with subtyping and type operators, is expressive enough to serve as a functional core calculus for typed object-oriented languages. The versions considered in the literature usually support pointwise subtyping of type operators, only, where two types $S\ U$ and $T\ U$ are in subtype relation, if S and T are. In the widely cited, unpublished note [Car90], Cardelli presents $F^{\omega}_{\leq}$ in a more general form going beyond pointwise subtyping of type applications in distinguishing between monotone and antimonotone operators. Thus, for instance, $T\ U_1$ is a subtype of $T\ U_2$, if $U_1 \leq U_2$ and if T is a monotone operator.

My thesis extends $F^{\omega}_{\leq}$ by polarized application, it explores its proof theory, establishing decidability of polarized $F^{\omega}_{\leq}$. The inclusion of polarized application rules leads to an interdependence of the subtyping and the kinding system. This contrasts with pure $F^{\omega}_{\leq}$, where subtyping depends on kinding, but not vice versa. To retain decidability of the system, the equal-bounds subtyping rule for all-types is rephrased in the polarized setting as a mutual-subtype requirement of the upper bounds. To my knowledge, this is the first proof-theoretic account of a polarized extension of pure $F^{\omega}_{\leq}$.

1.6 Structure of the thesis

Part I covers the material about pure $F^\omega_\leq$ without polarization. Part II contains the core of the thesis, extending the calculus of higher-order subtyping by polarity information and exploring its metatheory. Both parts proceed largely in parallel and sometimes only the proof for the more complicated polarized calculus are included. Especially typing is considered only in the polarized setting. The concluding chapter in Part III discusses related work, and proposes directions for further research. Most of the proofs have been included in the appendix in Part V.

1.7 Publications

The material about pure $F^\omega_\leq$ in Part I has already been published together with Benjamin Pierce in the technical report [SP94] and in a journal version [PS97]. It has been included with minor rearrangements and some simplifications. The material of Part II has been presented at the annual Types-group workshop, "Subtyping, Inheritance, and Modular Development of Proofs" in Durham, 1997 and on the Kolloquium on "Programmiersprachen und Grundlagen der Programmierung" in Avendorf, Isle of Fehrmarn, 1997.

Part I

Higher-Order Subtyping

Chapter 2

Higher-Order Subtyping

This chapter introduces the syntax and the rules for kinding, subtyping, and typing of $F^{\omega}_{\leq}$ ("F-omega-sub"), a typed λ-calculus of polymorphic functions, subtyping, and type operators. To retain decidability of subtyping and typing, we present the calculus in its "Kernel"-variant, where the subtyping rule for universally quantified types relates only All-types with equal upper bounds (cf. Chapter 1).

$F^{\omega}_{\leq}$ is a typed λ-calculus supporting polymorphic function, subtyping, and type operators. Thus besides term abstraction $(\mathit{fun}(x{:}T)t)$ and application $(f\ a)$ of the simply typed λ-calculus and type abstraction $\mathit{fun}(A{:}K)t$ and application $(t\ T)$ of Girard's and Reynold's second-order polymorphic λ-calculus, it includes functions on the level of kinds, allowing abstraction $(\mathit{Fun}(A{:}K)T)$ and application $(T\ U)$ within type expressions. The basic typing judgement for $F^{\omega}_{\leq}$ is $\Gamma \vdash t \in T$, reads "term t has type T in context Γ," where Γ records the type of each free term variable x and the kind of each free type variable A.

To include subtyping, the declaration of each type variable A in contexts Γ is extended with an upper bound, written $A{\leq}T$, which constrains A to range only over subtypes of T in the appropriate kind. To allow new constraints of this form to be introduced into the context, the universal quantifier $\mathit{All}(A{:}K)U$ is extended to a bounded quantifier $\mathit{All}(A{\leq}T)U$.

To ensure that the new system can still type all the terms of plain F^{ω}, we assume that the subtype relation in every kind K has a maximal element $\mathit{Top}(K)$. The assumption $A{\leq}\mathit{Top}(K)$ replaces $A{:}K$ in the contexts. For type operators of kind $K_1 \to K_2$, the subtype relation is just the *pointwise* extension of subtyping for K_2: a type operator $S \in K_1 \to K_2$ is smaller than $T \in K_1 \to K_2$ if $S\ U \leq T\ U$ for every $U \in K_1$.

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

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

Intuitively, a function f whose results inhabit S_2 whenever its arguments inhabit S_1 may safely be substituted for a function in $T_1 \to T_2$, provided that any element of T_1 that might be given as an argument to f can safely be used as an element of S_1 and that f's result, an element of S_2, can be used in place of the expected T_2. To retain decidability also in the higher-order case, we adopt the Kernel Fun subtyping rule for universal quantifiers, insisting on equal bounds for the two All-types under comparison:

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

That is, a polymorphic function $f \in \mathit{All}(A{\leq}U)S_2$ can be used in a context that expects an element of $\mathit{All}(A{\leq}U)T_2$, provided that, for each legal argument type T, the value of f at T can safely be used as an element of T_2. The complete definition of the subtyping system is presented in Section 2.5.

2.1 Syntax

The syntax of $F^{\omega}_{\leq}$ is given by the following abstract grammar.

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

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

The system allows to derive four families of statements: well-formedness of contexts, kinding, subtyping, and finally typing.

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

In the following, each set of statements will be inductively axiomatized by a corresponding set of derivation rules.

2.2 Contexts

Well-formed contexts are inductively defined by the three rules below. The empty context is well-formed, and a well-formed context can be extended by adding a declaration for a type variable or a term variable. The side condition concerning the free appearance of the variable assures that each variable can be declared at most once in a well-formed context.

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

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

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

For a term variable x declared in a well-formed context Γ, we write $\Gamma(x)$ for its type; likewise $\Gamma(A)$ for the upper bound of a type variable A, i.e., if $\Gamma = \Gamma_1,\, x{:}T,\, \Gamma_2$, then $\Gamma(x) = T$, and if $\Gamma = \Gamma'_1,\, A{\leq}T,\, \Gamma'_2$, then $\Gamma(A) = T$.

2.3 Conversion

With kind abstraction and application we need to consider reduction and conversion within types. As basic notion of reduction, we use a slight extension of the standard β-conversion relation: in addition to reduction steps of the usual form $(Fun(A{:}K_1)T)\ U \longrightarrow_\beta [U/A]T$, we introduce reductions of the form $Top(K_1 \rightarrow K_2)\ T \longrightarrow_\top Top(K_2)$, which relate the maximal elements of different kinds.

Definition 2.1 ($\beta\top$-reduction) The relation $\longrightarrow_{\beta\top}$ is the smallest binary relation on types satisfying the following rules:

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

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

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

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

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

The many-step $\beta\top$-reduction relation $\longrightarrow^{*}_{\beta\top}$ is the transitive closure of one-step reduction; $\beta\top$-equivalence, written $=_{\beta\top}$, is its transitive and symmetric closure; $\longrightarrow^{+}_{\beta\top}$ denotes reductions containing at least one proper β- or $\top$-step. When T has a normal form, it will necessarily be unique, we denote it by $T^{!}$. Reduction to $\beta\top$-normal form is written $\longrightarrow^{!}_{\beta\top}$.

2.4 Kinding

With abstraction and application on type level one has to check the appropriate usage of type operators by another level of inference rules, leading to what is called a three-level type system. Types take care of safe usage of terms, and the "types" of types, called *kinds*, control the safe application of type operators. The kinding system imposed as third level upon the types is fairly simple: basically the simply typed λ-calculus, with the base kind $\star$ classifying ordinary types (which are inhabited by terms), while kinds of the form $K_1 \to K_2$ classify *type operators*: functions mapping types of kind K_1 to types of kind K_2 (cf. K-Arrow-I and K-Arrow-E below). The proper types for functions and polymorphic functions carry the base kind $\star$. The subtyping system of the next section will axiomatize $\mathit{Top}(K)$ as maximal type in its kind K. The kind of type variables, finally, is determined by the kind of their upper bound.

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

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

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

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

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

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

2.5 Subtyping

The subtyping relation orders the types of each kind. The conversion rule S-Conv and transitivity (rule S-Trans) define $\leq$ an order relation on types, respecting $\beta\top$-equivalence, and with type $\mathit{Top}(K)$ as maximal type for each kind (rule S-Top). Type variables are smaller than their upper bound as declared the context Γ. As mentioned, function types behave contravariantly on the left-hand side of the arrow and covariantly on the right, and two All-types are compared by relating their bodies and by insisting on their upper bound to coincide. The pointwise application rule finally S-App lifts the subtype relation of two type operators to type applications. Correspondingly, the rule for type abstraction.

$$\frac{S =_{\beta\top} T \qquad \Gamma\ \vdash\ S, T\ \in\ K}{\Gamma\ \vdash\ S\ \leq\ T} \qquad \text{(S-Conv)}$$

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

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

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

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

$$\frac{\Gamma, A{:}K_1 \vdash S \leq T}{\Gamma \vdash \mathit{Fun}(A{:}K_1)S \leq \mathit{Fun}(A{:}K_1)T} \qquad \text{(S-Abs)}$$

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

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

2.6 Typing

The typing relation $\Gamma \vdash t \in T$ is a straightforward extension of the one for $F_{\leq}$. The rule T-Subsumption captures the intended interpretation of subtyping as “safe substitutability” and connects the subtyping with the typing system. The only axiom of the typing system relates term variables with their declaration in the context. Arrow- and All-types possess a pair of introduction- and elimination rules, each. In the first case by term abstraction and term application from the simply typed λ-calculus, for universally quantified types, correspondingly polymorphic type abstraction and type application.

$$\frac{\Gamma \vdash s \in S \quad \Gamma \vdash T \in \star \quad \Gamma \vdash S \leq T}{\Gamma \vdash s \in T} \qquad \text{(T-Subsumption)}$$

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

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

$$\frac{\Gamma \vdash f \in T_1 \rightarrow T_2 \qquad \Gamma \vdash a \in T_1}{\Gamma \vdash f\ a \in T_2} \qquad \text{(T-Arrow-E)}$$

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

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

Chapter 3

Decidability of $F^{\omega}_{\leq}$

This chapter proves decidability of subtyping for pure $F^{\omega}_{\leq}$, i.e., we develop a deterministic decision procedure that, given two types S and T together with a context, answers the question whether in that context S is smaller than T. The problem for pure $F^{\omega}_{\leq}$ is simpler than for polarized $F^{\omega}_{\leq}$ in Chapter 5 in the main part of the thesis, but some of the difficulties we will face later in the polarized setting are already discussed here. Thus we try to proceed in parallel in both cases, and the proofs for $F^{\omega}_{\leq}$, if included at all, can be found in Chapter B in the appendix.

The material of this chapter has been published together with Benjamin Pierce in [PS97]. It has been included with some rearrangement and minor simplifications.

3.1 Introduction

We start with a discussion of the main difficulties to obtain a subtyping algorithm for deciding for two types, whether they are in subtype relation wrt. to a given context.

Considering the subtype rules from Section 2.5, we identify two major sources of non-determinism standing in the way for an algorithm: the possibility to convert types into $\beta\top$-equivalent ones, expressed in rule S-Conv, and the rule of transitivity, also called the "cut-rule".

Conversion and reduction

The conversion rule S-Conv allows to convert a type to $\beta\top$-equivalent ones. For an algorithm we cannot tolerate such a liberty, that in order to find out whether $S \leq T$ we have to check all, which is infinitely many, $\beta\top$-equivalent pairs of types. The usual and obvious approach [AC96b] [Com95b] [AC96c] [Che96] [PS97] to this problem is to prove that it suffices to check the subtype relation for the unique normal forms of types, only.

As a first step towards a system operating exclusively on types in normal form we use a directed version of the system by distributing the effect of the undirected conversion rule over the rest of the subtype rules. This means that for each rule we allow the types of the goal to be reduced by an arbitrary number of steps in the premises. For example, instead of S-Arrow we will use the following rule:

$$\frac{S \longrightarrow^{*}_{\beta\top} S_1 \to S_2 \qquad T \longrightarrow^{*}_{\beta\top} T_1 \to T_2 \qquad \Gamma \vdash T_1 \leq S_1 \qquad \Gamma \vdash S_2 \leq T_2}{\Gamma \vdash S \leq T}$$

Transitivity and promotion

The second major obstacle mentioned is the rule of transitivity

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

which, proceeding from the goal to the subgoals amounts to guess the cut-type U. Aiming for an algorithm we cannot live with this kind of non-determinism and our intention is to prove S-Trans superfluous by a cut-elimination argument. So the question is: can we eliminate R-Trans? The answer is, almost. It is well-known already from the second-order case of $F_{\leq}$ that there are cases where the derivability of a statement essentially depends on transitivity. Statements with variables on the left-hand side cannot, in general, be proved without using transitivity. For example, suppose $\Gamma = C{:}\star\,, B{\leq}C{:}\star\,, A{\leq}B{:}\star$. Then the statement $\Gamma \vdash A \leq C \in \star$ can only be derived using S-Trans:

$$\dfrac{\dfrac{}{\Gamma \vdash A \leq B}\,\text{S-TVar} \qquad \Gamma \vdash B \leq C}{\Gamma \vdash A \leq C}\,\text{S-Trans}$$

The instance of transitivity here connects A's the upper bound B with type C, where A and B are related by the variable axiom S-TVar. This means nothing lies between a type variable of A and its upper bound, thus we can take care of transitivity by internalizing S-TVar and S-Trans into one rule:

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

Together with S-Conv this rule subsumes the axiom for type variables S-TVar. Now writing $A \uparrow_\Gamma \Gamma(A)$ for "the type variable A *promotes* to its upper bound $\Gamma(A)$ from the context." we can reformulate the enriched variable subtyping rule S-TVar-Plus-Trans as

$$\frac{A \uparrow_\Gamma \Gamma(A) \qquad \Gamma \ \vdash \ \Gamma(A) \ \leq \ T}{\Gamma \ \vdash \ A \ \leq \ T}$$

or, more generally

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

where the partial function $\uparrow_\Gamma$ is undefined except on type variables.

With these instances of transitivity taken care of, one can use standard induction to show that all other instances can be eliminated. That this elimination is possible means that nothing lies in between a type A and its upper bound $\Gamma(A)$. This gives rise to a subtyping algorithm (or a semi-decision procedure, depending on the form of All-rule one allows) for the second order fragment $F_{\leq}$ [CG92] [CG90] [BCGŠ91].

In the presence of higher-order subtyping, we encounter one new kind of situation in which transitivity plays an essential role. For example, in the context

$$\Gamma = A{:}\star\,, F{\leq}Id_{\star\to\star},$$

where $Id_{\star\to\star} = Fun(B{:}\star)B$, the statement $\Gamma \vdash F\ A \leq A$ is provable as follows:

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash F \leq Id_{\star\to\star}}\,\text{S-TVar}}{\Gamma \vdash F\ A \leq Id_{\star\to\star}\ A}\,\text{S-App} \qquad \dfrac{}{\Gamma \vdash Id_{\star\to\star}\ A \leq A}\,\text{S-Conv}}{\Gamma \vdash F\ A \leq A}\,\text{S-Trans}$$

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

Definition 3.1 (Promotion) Assume type $A\ T_1 \dots T_n \in K$ well-kinded in context Γ. The *promotion* of type $A\ T_1 \dots T_n$ in the context Γ, written $A\ T_1 \dots T_n \uparrow_\Gamma \Gamma(A)\ T_1 \dots T_n$, is defined as $\Gamma(A)\ T_1 \dots T_n$.

With this definition we can include the following rule into our system, to capture the essential uses of transitivity for type variables and the application of type variables.

$$\frac{S \longrightarrow^*_{\beta\top} \uparrow_\Gamma U \qquad \Gamma \vdash U \leq T}{\Gamma \vdash S \leq T} \qquad \text{(R-Promote)}$$

Proof outline

As said, the two major obstacles for a deterministic decision procedure are type *conversion* and the rule of *transitivity*. We address neither of them in the original system of Section 2.5, but in a directed version of the subtyping system, which we will call *reducing system*, with the aim to prove that no expressivity is lost by *eliminating* the cut rule and when the arbitrary reduction sequences in the premises are replaced by *normalizing* reductions. The key to the proof that we can indeed work on types in normal form will be a subject reduction property, i.e., preservation of subtyping under reduction.

Before we turn to subtyping, we start in Section 3.2 dealing with the more primitive kinding statements. In Section 3.3 we present the mentioned directed variant of the subtyping system. We proceed with the proof of subject reduction and cut-elimination in Section 3.4 and 3.5. Finally Section 3.6 contains the decision procedure for $F^\omega_\leq$ together with the proof of soundness and completeness.

3.2 Kinding

The task of this section is to develop a kinding algorithm, showing decidability of kinding for $F^\omega_\leq$. The second main point is subject reduction for kinding, which we will need at different places in the coming proofs for the subtyping system. First we collect some standard properties of the kinding system.

3.2.1 Properties of the kinding system

Lemma 3.2 (Transposition and weakening) Assume $A_2 \notin \mathit{fv}(T_1)$ and let Γ abbreviate the context $\Gamma_1,\ A_2{\leq}T_2,\ A_1{\leq}T_1,\ \Gamma_2$. Let further Γ' be be a well-formed extension of $\Gamma_1,\ A_1{\leq}T_1,\ A_2{\leq}T_2,\ \Gamma_2$. If $\Gamma \vdash S \in K$, then $\Gamma' \vdash S \in K$.

Lemma 3.3 (Context update for kinding) If $\Gamma_1,\ A{\leq}S,\ \Gamma_2 \vdash T \in K$ and $\Gamma_1 \vdash S, S' \in K'$, then $\Gamma_1,\ A{\leq}S',\ \Gamma_2 \vdash T \in K$.

Lemma 3.4 (Generation for contexts)

1. If $\vdash \Gamma\ \mathit{ok}$, then:
   (a) $\Gamma = \bullet$; or
   (b) $\Gamma = \Gamma',\ x{:}T$ with $\vdash \Gamma'\ \mathit{ok}$ and $\Gamma' \vdash T \in \star$ as subderivation; or
   (c) $\Gamma = \Gamma',\ A{\leq}T$ with $\vdash \Gamma'\ \mathit{ok}$ and $\Gamma' \vdash T \in K$ as subderivations.
2. If $\Gamma \vdash T \in K$, then $\vdash \Gamma\ \mathit{ok}$ as subderivation.

Lemma 3.5 (Generation for kinds)

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

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

Lemma 3.6 (Uniqueness of kinding) If $\Gamma \vdash T \in K$ and $\Gamma \vdash T \in K'$, then $K = K'$.

This justifies the following notation:

Definition 3.7 The unique kind of a well-kinded type T in a context Γ is written $\mathit{kind}_\Gamma T$.

3.2.2 Kinding algorithm

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

Definition 3.8 (Algorithmic kinding) The *algorithmic kinding* relation $\Gamma \vdash_{\mathcal{A}} T \in K$ is the smallest relation closed under rules of Section 2.4, with rule K-TVAR replaced by the following rule:

$$\frac{\Gamma_1 \vdash_{\mathcal{A}} T \in K \qquad \vdash_{\mathcal{A}} \Gamma_1, A{\leq}T, \Gamma_2\ ok}{\Gamma_1, A{\leq}T, \Gamma_2 \vdash_{\mathcal{A}} A \in K} \qquad \text{(K-TVAR-A)}$$

Lemma 3.9 (Context strengthening for algorithmic kinding)

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

Lemma 3.10 (Decidability of kinding) The relations $\vdash \Gamma\ ok$ and $\Gamma \vdash T \in K$ are decidable.

3.2.3 Subject reduction for kinding

In this section we will prove subject reduction, i.e., preservation of kinding under reduction (Lemma 3.14 on the facing page). With substitution as underlying mechanism, we first need preservation of kinding under substitution.

Lemma 3.11 (Transposition and weakening) Let Γ abbreviate the context $\Gamma_1, A_2{\leq}T_2, A_1{\leq}T_1, \Gamma_2$, with $A_2 \notin fv(T_1)$. Let further Γ' be be a well-formed extension of $\Gamma_1, A_1{\leq}T_1, A_2{\leq}T_2, \Gamma_2$. If $\Gamma \vdash_{\mathcal{A}} S \in K$, then $\Gamma' \vdash_{\mathcal{A}} S \in K$.

Lemma 3.12 (Top reduction) If $\Gamma \vdash\ Top(K)\ T_1 \ldots T_n \in K'$, then there exists a kind K' such that $Top(K)\ T_1 \ldots T_n \longrightarrow^{*}_{\beta\top} Top(K')$.

Lemma 3.13 (Substitution preserves kinding) Assume $\Gamma_1 \vdash U \in K$. Let Γ abbreviate the context $\Gamma_1, A'{:}K, \Gamma_2$ and Γ' abbreviate $\Gamma_1, [U/A']\Gamma_2$.

1. If $\vdash \Gamma\ ok$, then $\vdash \Gamma'\ ok$.
2. If $\Gamma \vdash T \in K'$, then $\Gamma' \vdash [U/A']T \in K'$.

We can use preservation of kinding under substitution to prove subject reduction, and further invariance under conversion.

Lemma 3.14 (Subject reduction for kinding)

1. If $\vdash \Gamma$ *ok* and $\Gamma \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} \Gamma'$, then $\vdash \Gamma'$ *ok*.
2. If $\Gamma \vdash S \in K$ and $S \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} T$ with $\Gamma \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} \Gamma'$, then $\Gamma' \vdash T \in K$.[1]

Corollary 3.15 (Kind invariance under conversion) Assume T' well-kinded in Γ. If $T' =_{\beta\top} T$ and $\Gamma \vdash T \in K$, then $\Gamma \vdash T' \in K$.

3.3 The reducing system

As mentioned in the introductory Section 3.1, as a first step towards an algorithm, we dispense with the rule of conversion, distributing its effect over the rest of the rules. This is done by allowing the types to be arbitrarily reduced in the premises. The rule of promotion R-PROMOTE generalizes the rule for type variables and takes care of the essential uses of transitivity implicit in T-TVAR. This leads to the following variant of the subtyping system.

Definition 3.16 (Reducing system) The *reducing system* is inductively given by the following set of rules:

$$\frac{\Gamma \vdash S \leq U \qquad \Gamma \vdash U \leq T \qquad \Gamma \vdash U \in K}{\Gamma \vdash S \leq T} \quad \text{(R-Trans)}$$

$$\frac{S \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} U \qquad T \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} U}{\Gamma \vdash S \leq T} \quad \text{(R-Refl)}$$

$$\frac{S \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} U \uparrow_\Gamma U' \qquad \Gamma \vdash U' \leq T}{\Gamma \vdash S \leq T} \quad \text{(R-Promote)}$$

$$\frac{\Gamma \vdash S \in K \qquad T \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} \mathit{Top}(K)}{\Gamma \vdash S \leq T} \quad \text{(R-Top)}$$

$$\frac{\begin{array}{cc} S \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} S_1 \to S_2 & T \longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top} T_1 \to T_2 \\ \Gamma \vdash T_1 \leq S_1 & \Gamma \vdash S_2 \leq T_2 \end{array}}{\Gamma \vdash S \leq T} \quad \text{(R-Arrow)}$$

[1]The relation $\longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top}$ on contexts is the obvious extension of $\beta\top$-relation from types to contexts.

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

$$\frac{S \longrightarrow^*_{\beta\top} Fun(A{:}K)S' \quad T \longrightarrow^*_{\beta\top} Fun(A{:}K)T' \qquad \Gamma, A{:}K \vdash S' \leq T'}{\Gamma \vdash S \leq T \in} \quad \text{(R-Abs)}$$

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ U \qquad T \longrightarrow^*_{\beta\top} T_1\ U \qquad \Gamma \vdash S_1 \leq T_1}{\Gamma \vdash S \leq T} \quad \text{(R-App)}$$

Notation 3.17 To avoid confusion, we distinguish derivations in different systems by marking the turnstile symbol: $\vdash_{\mathcal{O}}$ for the original system, $\vdash_{\mathcal{R}}$ for the reducing system, $\vdash_{\mathcal{S}}$ for strong derivations in the reducing system, $\vdash_{\mathcal{C}}$ for cut-free derivations in the reducing system, and $\vdash_{\mathcal{CS}}$ for strong, cut-free derivations in the reducing system.

The system is just a variant of our original formulation and is nondeterministic since it contains transitivity and arbitrary reductions in the premises. So the tasks of the remainder of this chapter are the proof that we can replace the arbitrary reductions premises by normalizing reductions, that we can eliminate the cut rule, and finally, the development of a subtyping algorithm.

Before we start with subject reduction in the following section, we need to collect some technical properties of the reducing system.

Lemma 3.18 (Preservation of kinding under promotion) If $\Gamma \vdash T \in K$ and $T \uparrow_\Gamma T'$, then $\Gamma \vdash T' \in K$.

Lemma 3.19 (Expansion preserves subtyping) Assume S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{R}} S' \leq T'$ where $S \longrightarrow^*_{\beta\top} S'$ and $T \longrightarrow^*_{\beta\top} T'$, then $\Gamma \vdash_{\mathcal{R}} S \leq T$. The same is correspondingly true for cut-free and strong derivations.

Next we prove a simple invariant of the system, namely that starting with two well-kinded types, well-kindedness is preserved for all subderivations.

Lemma 3.20 (Well-kindedness of subderivations) Suppose the types S and T well-kinded in Γ. Let d be a derivation of $\Gamma \vdash_{\mathcal{R}} S \leq T$ and d' a derivation of $\Gamma' \vdash_{\mathcal{R}} S' \leq T'$. If d' is a subderivation of d, then $\Gamma' \vdash S' \in K'$ and $\Gamma' \vdash T' \in K'$ for some kind K'.

Lemma 3.21 (Maximality of *Top*) Assume the types $Top(K)\ S_1 \ldots S_n$ and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} Top(K)\ S_1 \ldots S_n \leq T$, then $T \longrightarrow^*_{\beta\top} Top(K')$ for some kind K'.

3.4 Subject reduction for subtyping

Next we show that, for cut-free derivations, $\beta\top$-reduction in types does not interfere with the subtyping judgment. This will allow us to carry out the cut-elimination proof of in the system with strong derivations so that we can rely on the subgoals being in unique normal form.

The cornerstone of the argument is the preservation of the subtype relation under substitution.

Lemma 3.22 (Substitution preserves subtyping) Let Γ stand for the context $\Gamma_1, A{:}K, \Gamma_2$ and Γ' abbreviate $\Gamma_1, [U/A]\Gamma_2$. Suppose further $\Gamma_1 \vdash U \in K$, and S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma' \vdash_{\mathcal{C}} [U/A]S \leq [U/A]T$.

The following two lemmas will allow us to perform one outermost β-reduction step, resp. one $\top$-reduction step, on both sides of a subtyping statement. The lemma for the outermost β-step builds upon preservation of subtyping under substitution.

Lemma 3.23

1. Assume the types $S\ U$ and $(\mathit{Fun}(A{:}K)T)\ U$ well-kinded in context Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T$, then $\Gamma \vdash_{\mathcal{C}} S\ U \leq [U/A]T$.

2. Assume the types $(\mathit{Fun}(A{:}K)S)\ U$ and $T\ U$ well-kinded in context Γ. If $\Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S \leq T$, then $\Gamma \vdash_{\mathcal{C}} [U/A]S \leq T\ U$.

Lemma 3.24 Assume the types $\mathit{Top}(K')\ U$ and T well-kinded in Γ, and assume further $\mathit{Top}(K')\,U \longrightarrow_{\top} \mathit{Top}(K'')$. If $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K') \leq T$, then $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K'') \leq T\,U$.

Lemma 3.25 (Outer β-step) Assume the two types S and $(\mathit{Fun}(A{:}K)T)\ U$ well-kinded in context Γ.

1. If $\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K)T)\ U$, then $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T$.

2. If $\Gamma \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K)T)\ U \leq S$, then $\Gamma \vdash_{\mathcal{C}} [U/A]T \leq S$.

It is now easy to generalize preservation of subtyping under one outer reduction step to an arbitrary parallel reduction step. In the lemma we use parallel reduction of contexts (written $\Gamma \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top}\Gamma'$) as the pointwise extension of parallel reduction on types.

Lemma 3.26 (Parallel reduction preserves subtyping) Assume the types S and T well-kinded in Γ.

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

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

After we can do one parallel step, the generalization to arbitrary reduction sequences is a direct corollary.

Corollary 3.27 (Reduction preserves subtypes) Assume the types S and T well-kinded in Γ. If $S \longrightarrow^{*}_{\beta\top} S'$ and $T \longrightarrow^{*}_{\beta\top} T'$ with $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma \vdash_{\mathcal{C}} S' \leq T'$.

Corollary 3.28 Suppose the four types S, S', T, and T' well-kinded in Γ with $S =_{\beta\top} S'$ and $T =_{\beta\top} T'$. If $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma \vdash_{\mathcal{C}} S' \leq T'$.

Now that we know that subtyping in cut-free derivations is preserved under reduction we can use this fact to prove that a cut-free derivation can be strengthened, i.e., turned into a derivation where all subtyping rules reduce their subgoals to normal forms.

Lemma 3.29 (Strengthening) Suppose S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq T$, then $\Gamma \vdash_{\mathcal{CS}} S \leq T$.

Proof of Lemma 3.29: We are given $\Gamma \vdash_{\mathcal{C}} S \leq T$. By Corollary 3.27 we obtain $\Gamma \vdash_{\mathcal{C}} S^{!} \leq T^{!} \in K$. The derivation of this statement is not necessarily strong since the rule of promotion may generate subgoals not in normal form. By well-kindedness of subderivation one can normalize these again using the same corollary.

The fact that substitution into a subtyping statement (Lemma 3.22) does not increase the number of instances of R-Promote in the subtyping derivation, implies that neither $\beta\top$-reduction increases this number, which guarantees that this process of strengthening terminates. $\square$

3.5 Cut elimination

We are now ready we prove that admissibility of S-Trans for strong-cut-free derivation, i.e., we will be prove that

$$\Gamma \vdash_{\mathcal{CS}} S \leq U \qquad \text{and} \qquad \Gamma \vdash_{\mathcal{CS}} U \leq T$$

by two cut-free derivations implies that

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

can be derived without cut (assuming U well-kinded in Γ). The proof of this implication will be carried out as induction over the combined length of derivations of $\Gamma \vdash_{\mathcal{CS}} S \leq U$ and $\Gamma \vdash_{\mathcal{CS}} U \leq T$ (cf. also [BCGŠ91], [CG92], [CMMŠ94], or [Com95b]).

If, for instance, the derivations of the left and the right statement both end with an instance of R-ARROW, i.e., we are given $S = S_1 \to S_2$, $U = U_1 \to U_2$, and $T = T_1 \to T_2$, we obtain four subgoals: $\Gamma \vdash_{\mathcal{CS}} U_1 \leq S_1$ and $\Gamma \vdash_{\mathcal{CS}} S_2 \leq U_2$, together with $\Gamma \vdash_{\mathcal{CS}} T_1 \leq U_1$ and $\Gamma \vdash_{\mathcal{CS}} U_2 \leq T_2$.[2] From the pair of the first and the third statement induction yields $\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1$; analogously, the remaining two statements give $\Gamma \vdash_{\mathcal{CS}} S_2 \leq T_2$, and thus by the subtyping rule for arrow types:

$$\frac{\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1 \qquad \Gamma \vdash_{\mathcal{CS}} S_2 \leq T_2}{\Gamma \vdash_{\mathcal{CS}} S_1 \to S_2 \leq T_1 \to T_2}$$

All possible combinations of subtyping rules for the left and the right derivation can either be solved directly — for instance, if one of the rules is an instance of reflexivity — or by the inductive argument sketched in the case of arrow types. The induction works especially also for universally quantified types, as both $\Gamma \vdash_{\mathcal{CS}} \mathit{All}(A{\leq}U_1)S_2 \leq \mathit{All}(A{\leq}U_1)U_2$ and $\Gamma \vdash_{\mathcal{CS}} \mathit{All}(A{\leq}U_1)U_2 \leq \mathit{All}(A{\leq}U_1)T_2$ extend the context Γ by the same bound $A{\leq}U_1$.

The notable exception concerns the rule of promotion: if the statement on the right $\Gamma \vdash_{\mathcal{CS}} U \leq T$ produces as subgoal $\Gamma \vdash_{\mathcal{CS}} U' \leq T$ with $U = A\, U_1 \ldots U_n \uparrow_\Gamma U'$, the inductive argument fails. The problem lies in the asymmetry of the promotion rule in that it affects only the left-hand side of a subtype statement. There is no possibility to make direct use of the subderivation for $\Gamma \vdash_{\mathcal{CS}} U' \leq T$ for the inductive argument, as the combined length of the derivations of $\Gamma \vdash_{\mathcal{CS}} S \leq U$ and $\Gamma \vdash_{\mathcal{CS}} U' \leq T$ may be smaller than the combined length of the original derivations, but we do no longer have a common cut-type.

But now we can profit by the restrictions we imposed upon the derivations. That the type U can be promoted means $U \longrightarrow^!_{\beta\top} A\, U_1 \ldots U_n$. Furthermore, in absence of transitivity, for a derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq U$, only reflexivity, the rule of application, and the promotion rule are available. This means that there exists a derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq U$, ending in a sequence of R-PROMOTE's preceded by reflexivity:

[2]For sake of illustration, assume the types involved to be in normal form already. Since we can work with strong derivations, the argument does not depend on this simplification.

$$\dfrac{S \longrightarrow^!_{\beta\top} \uparrow_\Gamma S_1 \qquad \dfrac{\dfrac{S_{m-1} \longrightarrow^!_{\beta\top} \uparrow_\Gamma S_m \qquad \dfrac{\begin{array}{c} S_m \longrightarrow^!_{\beta\top} A\ U_1 \ldots U_n \quad U \longrightarrow^!_{\beta\top} A\ U_1 \ldots U_n \\ \Gamma \vdash A\ U_1 \ldots U_n \in K \end{array}}{\Gamma \vdash_{\mathcal{CS}} S_m \leq U}}{\vdots}}{\Gamma \vdash_{\mathcal{CS}} S_1 \leq U}}{\Gamma \vdash_{\mathcal{CS}} S \leq U}$$

Since we are given $\Gamma \vdash_{\mathcal{CS}} U \leq T$, we also know $\Gamma \vdash_{\mathcal{CS}} A\ U_1 \ldots U_n \leq T$, and we can use the derivation above to construct the desired one for $\Gamma \vdash_{\mathcal{CS}} S \leq T$:

$$\dfrac{S \longrightarrow^!_{\beta\top} \uparrow_\Gamma S_1 \qquad \dfrac{\dfrac{S_{m-1} \longrightarrow^!_{\beta\top} \uparrow_\Gamma S_m \qquad \dfrac{S_m \longrightarrow^!_{\beta\top} A\ U_1 \ldots U_n \quad \Gamma \vdash_{\mathcal{CS}} A\ U_1 \ldots U_n \leq T}{\Gamma \vdash_{\mathcal{CS}} S_m \leq U}}{\vdots}}{\Gamma \vdash_{\mathcal{CS}} S_1 \leq T}}{\Gamma \vdash_{\mathcal{CS}} S \leq T}$$

Before we turn to the cut-elimination proof we cast this observation into a Lemma (3.33) which characterizes the derivations for subtypes of applications and which covers the critical case of promotion in the proof of cut-elimination. To describe the effect of a couple of instances of R-Promote upon a type, we introduce the following notation.

Definition 3.30 (Promotion and reduction) The binary relation $\nearrow_\Gamma$ between types is defined as $\longrightarrow^!_{\beta\top} \uparrow_\Gamma$. By $\nearrow^*_\Gamma$ we denote the corresponding multi-step-relation.

The following lemma expresses an easy reduction property of this relation.

Lemma 3.31 Assume the type $S\ T$ well-kinded in Γ where S and T in normal form. If $S \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\ S_1 \ldots S_n$, then $S\ T \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\ S_1 \ldots S_n\ T$.

Lemma 3.32 Suppose the types S, S', and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{CS}} S \leq T$ and $S' \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} S$, then $\Gamma \vdash_{\mathcal{CS}} S' \leq T$.

Lemma 3.33 (Subtypes of an application) Assume the types S and T well-kinded in context Γ with $T \longrightarrow^!_{\beta\top} A\ T_1 \ldots T_n$ for some $n \geq 0$. If $\Gamma \vdash_{\mathcal{CS}} S \leq T$, then $S \nearrow^*_\Gamma \ \longrightarrow^!_{\beta\top} A\ T_1 \ldots T_n$.

We now can turn to the cut-elimination proof.

Proposition 3.34 (Cut elimination) Assume S, T, and U well-kinded in context Γ. If $\Gamma \vdash_{\mathcal{CS}} S \leq U$ and $\Gamma \vdash_{\mathcal{CS}} U \leq T$, then $\Gamma \vdash_{\mathcal{CS}} S \leq T$.

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

If the statement $\Gamma \vdash_{\mathcal{CS}} S \leq U$ on the left-hand side ends with an instance of reflexivity, we are given $S \longrightarrow^!_{\beta\top} V$ and $T \longrightarrow^!_{\beta\top} V$. Thus $\Gamma \vdash_{\mathcal{CS}} S \leq T$ is justified directly by the derivation of $\Gamma \vdash U \leq T$. Similarly, if $\Gamma \vdash_{\mathcal{CS}} U \leq T$ on the right ends with R-Refl in the last step. If $\Gamma \vdash_{\mathcal{CS}} U \leq T$ ends with R-Top, the case follows immediately by well-kindedness of subderivations and R-Top. If R-Top is the last rule for the left derivation, the case follows by maximality of *Top* (Lemma 3.21) and R-Top. The cases for R-Promote on the left-hand side can be solved by well-kindedness of subderivations, induction, and the promotion rule.

Case: R-Promote on the right:
$$\frac{U \longrightarrow^!_{\beta\top} A\ U_1\ \ldots\ U_n \uparrow_\Gamma U' \quad \Gamma \vdash_{\mathcal{CS}} U' \leq T}{\Gamma \vdash_{\mathcal{CS}} U \leq T}$$

By Lemma 3.33 $S \nearrow^*_\Gamma \ \longrightarrow^!_{\beta\top} A\ U_1 \ldots U_n$. Since $\Gamma \vdash_{\mathcal{CS}} U \leq T$ implies $\Gamma \vdash_{\mathcal{CS}} A\ U_1 \ldots U_n \leq T$, the result follows by Lemma 3.32.

The remaining combinations are can either be solved directly or simpler. We only show the case for universally quantified types.

Case R-All:

$$\frac{\begin{array}{c} S \longrightarrow^!_{\beta\top} All(A{\leq}U_1)S_2 \\ U \longrightarrow^!_{\beta\top} All(A{\leq}U_1)U_2 \\ \Gamma, A{\leq}U_1 \vdash S_2 \leq U_2 \end{array}}{\Gamma \vdash_{\mathcal{CS}} S \leq U} \qquad \frac{\begin{array}{c} T \longrightarrow^!_{\beta\top} All(A{\leq}U_1)T_2 \\ U \longrightarrow^!_{\beta\top} All(A{\leq}U_1)U_2 \\ \Gamma, A{\leq}U_1 \vdash U_2 \leq T_2 \end{array}}{\Gamma \vdash_{\mathcal{CS}} U \leq T}$$

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

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

□

3.6 A subtyping algorithm for $F^\omega_\leq$

Now the system with strong, cut-free derivation almost immediately give rise to an algorithm for the subtyping system of Section 2.5. The only choices, we still have to resolve, concern reflexivity, the rule for maximal types, and the possible conflict between the promotion rule on the one hand and the rule for application on the other.

The first two cases mentioned are treated by ordering the rules in such a way that the easy cases R-Refl and R-Top are always checked first. The conflict between R-App and R-Promote is resolved in favor of promotion: by Lemma 3.33, a subtype of an application can always be promoted until it coincides with this application. This means if the application rule is applicable, the algorithm might as well choose the promotion rule, which explains why the rules of Definition 3.35 can do without a rule A-App.

Definition 3.35 (Subtyping algorithm) Let the types S and T be well-kinded in context Γ and in normal form. The subtyping algorithm $\Gamma \vdash_{\mathcal{A}} S \leq T$ is inductively given by the rules below, applied on a first-match basis.

$$\Gamma \vdash_{\mathcal{A}} U \leq U \qquad \text{(A-Refl)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S \in K}{\Gamma \vdash_{\mathcal{A}} S \leq \mathit{Top}(K)} \qquad \text{(A-Top)}$$

$$\frac{S \uparrow_\Gamma \;\xrightarrow{!}_{\beta\top} U \qquad \Gamma \vdash_{\mathcal{A}} U \leq T}{\Gamma \vdash_{\mathcal{A}} S \leq T} \qquad \text{(A-Promote)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1 \leq S_1 \qquad \Gamma \vdash_{\mathcal{A}} S_2 \leq T_2}{\Gamma \vdash_{\mathcal{A}} S_1 \to S_2 \leq T_1 \to T_2} \qquad \text{(A-Arrow)}$$

$$\frac{\Gamma, A{\leq}U \vdash_{\mathcal{A}} S_2 \leq T_2}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A{\leq}U)S_2 \leq \mathit{All}(A{\leq}U)T_2} \qquad \text{(A-All)}$$

$$\frac{\Gamma, A{:}K \vdash_{\mathcal{A}} S' \leq T'}{\Gamma \vdash_{\mathcal{A}} \mathit{Fun}(A{:}K)S' \leq \mathit{Fun}(A{:}K)T'} \qquad \text{(A-Abs)}$$

Proposition 3.36 (Soundness and completeness) Assume the type S and T well-kinded in context Γ. Then $\Gamma \vdash_{\mathcal{O}} S \leq T$, iff. $\Gamma \vdash_{\mathcal{A}} S^! \leq T^!$.

Proposition 3.37 (Termination) Suppose S and T well-kinded in Γ and in normal form. Then the algorithm of Definition 3.35 always terminates with Γ, S, and T as input.

Proof: The assumption that the rules of the algorithm can generate a tree of subtyping statements of infinite depth contradicts strong termination (Lemma A.9) of $\longrightarrow_{\beta\top\Gamma}$-reduction from Definition A.10. □

Proposition 3.38 (Decidability of subtyping) The subtype relation $\Gamma \vdash S \leq T$ of $F^{\omega}_{\leq}$ is decidable.

Proof: In order to decide whether $\Gamma \vdash_{\mathcal{O}} S \leq T$, we can use the sound and complete (Proposition 3.36) algorithm Definition 3.35. Moreover, the algorithm always halts by Proposition 3.37. □

Part II

Polarized Higher-Order Subtyping

Chapter 4

Polarized $F^{\omega}_{\leq}$

This chapter extends the calculus $F^{\omega}_{\leq}$ from Chapter 2 by subtyping rules for a more general form of application, taking monotonicity information into account. Thus, besides the pointwise subtyping rule S-App for applications, the system now allows to derive for example $\Gamma \vdash T\ U_1 \leq T\ U_2$, if $\Gamma \vdash U_1 \leq U_2$ and provided T is monotone. Besides monotone operators, the systems formalizes also antimonotone and constant ones. In effect, $F^{\omega}_{\leq}$ appeared in a form including monotonicity information in Cardelli's note [Car90].

The inclusion of non-pointwise application into the subtyping system will affect the kinding system, as well, since determining the polarity of a type operator will be the task of the kinding system. What will make the investigation principally more challenging than for pure $F^{\omega}_{\leq}$ is the fact that the kinding system in turn will rest on subtyping derivations, leading to a mutual dependence of kinding and subtyping not present in this form in pure $F^{\omega}_{\leq}$.

4.1 Syntax

The syntax of polarized $F^{\omega}_{\leq}$ extends the one for the pure calculus from Section 2.1 by annotating arrow kinds with one of four different polarities. As the polarity of a type operator implies nothing about the polarity of its subtypes, we need the possibility to independently specify the polarity for bounded universal quantification: we thus extend the syntax of universal types from $All(A{\leq}T_1)T_2$ to $All(A{\leq}T_1{:}K_1)T_2$, containing polymorphic functions with type arguments smaller than T_1 and additionally of kind K_1. Correspondingly, the syntax of type abstraction for terms, and the form of the contexts for the declaration of bound type variables is adapted.

The syntax is given by the following abstract grammar.

$?$	$::=$	$\pm$	unspecified
	$\mid$	$+$	positive
	$\mid$	$-$	negative
	$\mid$	$\circ$	constant
K	$::=$	$\star$	kind of types
	$\mid$	$K \to^{?} K$	kind of type operators with polarity ?

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

In addition to the statements of pure $F^{\omega}_{\leq}$ — well-formedness of contexts, kinding, subtyping, and typing — the system contains statements for subkinding:

$K \leq K'$	K is a subkind of K'
$\vdash \Gamma\ ok$	Γ is a well-formed context
$\Gamma \vdash T \in K$	type T has kind K in context Γ
$\Gamma \vdash S \leq T \in K$	S is a subtype of T in Γ
$\Gamma \vdash t \in T$	term t has type T in Γ

Subtyping statements now take the form $\Gamma \vdash S \leq T \in K$, meaning that S is a subtype of T, both bearing kind K. This more general form is needed, as the system will no longer enjoy a unique kinding property. As usual, we consider types, terms, contexts, and statements only up-to α-equality.

4.2 Contexts

Besides extra kinding annotation for type variables, context-formation for polarized $F^{\omega}_{\leq}$ is the same as for pure $F^{\omega}_{\leq}$. The empty context is well-formed, and a well-formed context can be extended by adding a declaration for a new type or term variable.

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

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

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

For a type variable A appearing in a well-formed context Γ, we write $kind_\Gamma A$ for its kind as declared in Γ, i.e., if $\Gamma = \Gamma_1, A{\leq}T{:}K, \Gamma_2$, then $kind_\Gamma A = K$, and, as before, its upper bound $\Gamma(A) = T$.

4.3 Subtyping

The following two sections are concerned with the definition of the subtyping system and, mutually dependent, the rules for the kinding system. We start with the formalization of the subtype relation.

The difference between the $F^{\omega}_{\leq}$ and the polarized version of this section is most visible in the application rules. For $F^{\omega}_{\leq}$, the pointwise application rule lifts the subtype relation of two type operators S and T to type applications, as expressed in S-App:

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

To extend the system beyond pointwise subtyping we need a characterization of the monotonicity or *polarity* of type operators — this will be the task of the kinding system in the following section — and new subtyping rules exploiting the additional knowledge about the polarity of the operators.

To start with the monotone case, the basic intuition about a monotone type operator T is that, applied to two arguments in subtype ordering, the application is ordered likewise. This is captured by the following rule S-App+ (ignoring for the moment the additional kinding information in the subtyping statements for sake of simplicity):

$$\frac{\Gamma \vdash T\,+ \qquad \Gamma \vdash U_1 \leq U_2}{\Gamma \vdash T\,U_1 \leq T\,U_2}$$

The premise $\Gamma \vdash T\,+$ expresses monotonicity of the type operator T, abbreviating a kinding statement of the form $\Gamma \vdash T \in K_1 \rightarrow^{+} K_2$ for some kinds K_1 and K_2. For duality, we will have to consider antimonotone ones, as well, and we include a corresponding rule S-App−

$$\frac{\Gamma \vdash T\,- \qquad \Gamma \vdash U_1 \leq U_2}{\Gamma \vdash T\,U_2 \leq T\,U_1}$$

into the system.

Besides expressing that a type operator depends monotonely or anti-monotonely upon its arguments, we would like to be able to denote that it does not depend upon its arguments at all, i.e., that it is *constant* in its arguments. Using the symbol $\circ$ for constant polarity, the behavior of constant application is formalized by rule S-App$\circ$:

$$\frac{\Gamma \vdash T\,\circ}{\Gamma \vdash T\,U_1 \leq T\,U_2}$$

For an operator to be constant is equivalent to be monotone and antimonotone at the same time. This can be seen by the following informal argument: suppose $\Gamma \vdash T\,+$ and $\Gamma \vdash T\,-$, and assume two arbitrary arguments U_1 and U_2 of an appropriate kind. Thus we know $\Gamma \vdash U_1 \leq \mathit{Top}(K)$ as well as $\Gamma \vdash U_2 \leq \mathit{Top}(K)$, yielding both $\Gamma \vdash T\;U_1 \leq T\;\mathit{Top}(K)$ and $\Gamma \vdash T\;\mathit{Top}(K) \leq T\;U_2$, using monotonicity and anti-monotonicity of T respectively, which transitivity combines to $\Gamma \vdash T\;U_1 \leq T\;U_2$. (The reverse direction is immediate since constant operators will also be monotone and antimonotone by subsumption.)

It is well worth noticing that for a type operator being constant does not mean it throws away its arguments. In other words, $\Gamma \vdash T\,\circ$ does not imply $\beta\top$-equivalence of $T\ U_1$ and $T\ U_2$. The reason is that type variables can be declared as constant operators, but they cannot swallow their arguments. The need to differentiate between constant operators and those ignoring their arguments altogether contrasts with the work of Fisher [Fis96] and Hofmann and Pierce [HP95b] which also determine the polarity of type operators (in [Fis96] called row functions). Since there, only type variables of kind $\star$ are considered, type variables cannot be applied to other types, and both notions coincide.

To make the system symmetric, we add a fourth polarized application rule, more generous than pointwise application, in that it does not insist on its arguments being identical, but requiring only that each argument is a subtype of the other. Writing $\Gamma \vdash U_1 \gtrless U_2$ for such pairs of subtyping statements $\Gamma \vdash U_1 \leq U_2$ and $\Gamma \vdash U_2 \leq U_1$, we can write this last application rule S-App± as (ignoring again kinding):

$$\frac{\Gamma \vdash T \pm \qquad \Gamma \vdash U_1 \gtrless U_2}{\Gamma \vdash T\ U_1 \leq T\ U_2}$$

Having type variables act as constant type operators means that not only constant operators differ from the ones ignoring their arguments, but also that the notion of β-equivalence of two types S and T and the fact that the two types are in mutual subtype relationship are to be distinguished.[1] This distinction is not present in pure $F^{\omega}_{\leq}$ and neither in the "polarized" calculi of Fisher [Fis96] and Hofmann and Pierce [HP95a]. But this fourth rule neither appears in the higher-order setting of Cardelli [Car90].

The remaining rules are similar to the ones for pure $F^{\omega}_{\leq}$. The conversion rule S-Conv and transitivity S-Trans define $\leq$ an order relation on types, respecting $\beta\top$-equivalence (the relation $=_{\beta\top}$ was introduced in Section 2.3 Here we use the obvious transfer of this definition to the slightly adapted syntax for universal types). A type variable is smaller than its upper bound as declared in the context where additionally its kind has to conform to the kinding part of the subtyping statement. Each kind contains a maximal type by rule S-Top. Since in the presence of subkinding, the types will no longer enjoy a unique kinding property, we have to check also $Top(K')$ for respecting the required kind. Type variables are smaller than their upper bound as declared the context Γ. As before, the rule S-Arrow for function types behaves contravariantly on the left-hand side of the arrow, and covariantly on the right. The rule for type operators compares the bodies of the two operators and use the kinding system to determine their polarity.

Finally the rule for universally quantified types. For we want to retain decidability of the subtyping system, we would like to stick close to the decidable variant of $F^{\omega}_{\leq}$,

[1]Technically, the fourth application rule is needed to obtain preservation of subtyping under reduction.

which insists on equal upper bounds for the All-types. By the same arguments that made it necessary to introduce S-App±, we are led to relax the conditions upon the two bounds of universally quantified types, requiring that they must be mutually smaller than each other.[2]

The subtype relation is thus inductively given by the set of rules below.

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

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

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

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

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{Fun}(A{:}K_1)S, \mathit{Fun}(A{:}K_1)T \in K_1 \to^{?} K_2 \\ \Gamma, A{:}K_1 \vdash S \leq T \in K_2\end{array}}{\Gamma \vdash \mathit{Fun}(A{:}K_1)S \leq \mathit{Fun}(A{:}K_1)T \in K_1 \to^{?} K_2} \tag{S-Abs}$$

$$\frac{\Gamma \vdash S \leq T \in K_1 \to^{\pm} K_2 \qquad \Gamma \vdash U \in K_1}{\Gamma \vdash S\,U \leq T\,U \in K_2} \tag{S-App}$$

$$\frac{\Gamma \vdash T \in K_1 \to^{\circ} K_2 \qquad \Gamma \vdash U_1, U_2 \in K_1}{\Gamma \vdash T\,U_1 \leq T\,U_2 \in K_2} \tag{S-App∘}$$

$$\frac{\Gamma \vdash T \in K_1 \to^{+} K_2 \qquad \Gamma \vdash U_1 \leq U_2 \in K_1}{\Gamma \vdash T\,U_1 \leq T\,U_2 \in K_2} \tag{S-App+}$$

$$\frac{\Gamma \vdash T \in K_1 \to^{-} K_2 \qquad \Gamma \vdash U_2 \leq U_1 \in K_1}{\Gamma \vdash T\,U_1 \leq T\,U_2 \in K_2} \tag{S-App−}$$

$$\frac{\Gamma \vdash T \in K_1 \to^{\pm} K_2 \qquad \Gamma \vdash U_1 \lesseqgtr U_2 \in K_1}{\Gamma \vdash T\,U_1 \leq T\,U_2 \in K_2} \tag{S-App±}$$

[2]One can see it also from another perspective: requiring mutual subtype relationship of the upper bounds to obtain decidability means in the pure, non-polarized case two $\beta\top$-convertible upper bounds which, in the presence of a conversion rule S-Conv, amounts to the same as the equal-bounds requirement of Kernel Fun. Here, with more informative subtyping rules for application, mutual subtype relationship does not imply convertibility.

$$\frac{\Gamma \vdash T_1 \leq S_1 \in \star \qquad \Gamma \vdash S_2 \leq T_2 \in \star}{\Gamma \vdash S_1 \to S_2 \leq T_1 \to T_2 \in \star} \qquad \text{(S-Arrow)}$$

$$\frac{\Gamma \vdash S_1 \gtrless T_1 \in K_1 \qquad \Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \leq T_2 \in \star}{\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \leq \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star} \qquad \text{(S-All)}$$

4.4 Kinding

This section deals with the kinding statements, mutually dependent from the subtyping statements of the previous section. As we have seen, arrow kinds come decorated with polarities to express monotonicity information. For instance, the type operator

$$\mathit{Fun}(A{:}\star)\,\mathit{Top}(\star) \to A$$

will carry the kind $\star \to^{+} \star$ indicating that it operates monotonely on its arguments of kind $\star$. A type operator $\mathit{Fun}(A{:}K)T$ is monotone, if for all types U_1 and U_2 of the appropriate kind, $U_1 \leq U_2$ implies $[U_1/A]T \leq [U_2/A]T$. This is captured in the following rule:

$$\frac{\begin{array}{c}\Gamma, A{:}K_1 \vdash T \in K_2 \\ \Gamma, A_2{:}K_1, A_1{\leq}A_2{:}K_1 \vdash [A_1/A]T \leq [A_2/A]T \in K_2\end{array}}{\Gamma \vdash \mathit{Fun}(A{:}K_1)T \in K_1 \to^{+} K_2}$$

A similar definition in a second-order setting appears in Abadi and Cardelli [AC96b], Section 8.1, using only three different variances: covariant, contravariant, and "invariant" operators, where the last term corresponds to our $\pm$-polarity.[3] The rule is also reminiscent to the way, in which in subtyping systems with recursion [AC93] the monotonicity of the recursion types under comparison is assured.

Following the discussion of the subtyping section we will use four polarities: monotone and antimonotone, constant, and a fourth one denoting absence of information. Intuitively, the polarities are ordered, e.g., a constant operator uses its arguments monotonely, as well. The order is summarized in the diagram below:[4]

[3]The different terminology is caused by a different perspective: Abadi and Cardelli [AC96b] would call a type operator of $\pm$-polarity "invariant" with the rationale, that it is sound to use such an operator only on invariant arguments. As far as $+$ and $-$ are concerned, there is no ambiguity.

[4]Also Fisher [Fis96] uses four polarities, again written somewhat differently and without distinction between constant polarity $\circ$ and the fact that a variable does not occur freely.

$$\begin{tikzcd}
& \pm & \\
+ \arrow[ur] & & - \arrow[ul] \\
& \circ \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

The intuitive ordering will be reflected in the kinding system by the fact that to derive a weaker polarity of an operator on has to prove stronger premises in the kinding rule.

The corresponding reflexive and transitive relation on polarities is written $? \leq ?'$. In accordance with usual conventions, the smaller the polarity, the better, so $\circ \leq -$ for example. In this way, the polarities form a small lattice and we use $? \vee ?'$ as notation for the binary least upper bound of $?$ and $?'$, and $? \wedge ?'$ for the greatest lower bound on this domain.

Now the definition of the kinding system is straightforward. First we use the $\leq$-relation on polarities just defined to impose a corresponding ordering upon kinds (for which we will once again use the symbol $\leq$). This ordering is formalized the axiom K-Refl and the rule K-Sub:

$$K \leq K \qquad \text{(K-Refl)}$$

$$\frac{K_1' \leq K_1 \qquad K_2 \leq K_2' \qquad ? \leq ?'}{K_1 \rightarrow^{?} K_2 \leq K_1' \rightarrow^{?'} K_2'} \qquad \text{(K-Sub)}$$

Under this ordering, we get subsumption on the level of kinds, expressed in rule K-Subsumption. The kind of a type variable A is determined by its declaration $\mathit{kind}_\Gamma A$ in the context Γ. The type $\mathit{Top}(K)$ is intended as the maximal type for the respective kind K. Since, regardless of type S, we think of the application $\mathit{Top}(K_1 \rightarrow^{?} K_2)\ S$ as bigger than all types of the appropriate kind K_2, the operator $\mathit{Top}(K_1 \rightarrow^{?} K_2)$ behaves constantly on its arguments; hence its kind $K_1 \rightarrow^{\circ} K_2$. This also justifies the following notation for contexts: we abbreviate $\Gamma_1, A{\leq}\mathit{Top}(K){:}K, \Gamma_2$ by $\Gamma_1, A{:}K, \Gamma_2$, as the maximal type $\mathit{Top}(K)$ carries no further formation.

The four rules for arrow-introduction establish the connection between the subtyping system and the kinding rules: the polarity of a type operator is determined by a subtyping derivation, with the appropriate assumption about the formal parameters in the context. As in pure $F^\omega_\leq$, arrow- and All-types for functions, respectively polymorphic functions finally carry the unique kind $\star$.

$$\frac{K' \;\leq\; K \qquad \Gamma \vdash T \;\in\; K'}{\Gamma \vdash T \;\in\; K} \qquad \text{(K-S\textsc{ubsumption})}$$

$$\frac{\vdash \Gamma \; ok}{\Gamma \vdash A \;\in\; kind_\Gamma A} \qquad \text{(K-TV\textsc{ar})}$$

$$\frac{\vdash \Gamma \; ok}{\Gamma \vdash Top(\star) \;\in\; \star} \qquad \text{(K-T\textsc{op}}\star\text{)}$$

$$\frac{\Gamma \vdash Top(K_2) \;\in\; K_2'}{\Gamma \vdash Top(K_1 \to^{?} K_2) \;\in\; K_1 \to^{\circ} K_2'} \qquad \text{(K-T\textsc{op})}$$

$$\frac{\Gamma, A{:}K_1 \vdash T \;\in\; K_2}{\Gamma \vdash Fun(A{:}K_1)T \;\in\; K_1 \to^{\pm} K_2} \qquad \text{(K-A\textsc{rrow}-I}\pm\text{)}$$

$$\frac{\begin{array}{c}\Gamma, A{:}K_1 \vdash T \;\in\; K_2 \\ \Gamma, A_2{:}K_1, A_1{\leq}A_2{:}K_1 \vdash [A_1/A]T \;\leq\; [A_2/A]T \;\in\; K_2\end{array}}{\Gamma \vdash Fun(A{:}K_1)T \;\in\; K_1 \to^{+} K_2} \qquad \text{(K-A\textsc{rrow}-I+)}$$

$$\frac{\begin{array}{c}\Gamma, A{:}K_1 \vdash T \;\in\; K_2 \\ \Gamma, A_2{:}K_1, A_1{\leq}A_2{:}K_1 \vdash [A_2/A]T \;\leq\; [A_1/A]T \;\in\; K_2\end{array}}{\Gamma \vdash Fun(A{:}K_1)T \;\in\; K_1 \to^{-} K_2} \qquad \text{(K-A\textsc{rrow}-I}-\text{)}$$

$$\frac{\begin{array}{c}\Gamma, A{:}K_1 \vdash T \;\in\; K_2 \\ \Gamma, A_1{:}K_1, A_2{:}K_1 \vdash [A_1/A]T \;\leq\; [A_2/A]T \;\in\; K_2\end{array}}{\Gamma \vdash Fun(A{:}K_1)T \;\in\; K_1 \to^{\circ} K_2} \qquad \text{(K-A\textsc{rrow}-I}\circ\text{)}$$

$$\frac{\Gamma \vdash S \;\in\; K_1 \to^{?} K_2 \qquad \Gamma \vdash T \;\in\; K_1}{\Gamma \vdash S\; T \;\in\; K_2} \qquad \text{(K-A\textsc{rrow}-E)}$$

$$\frac{\Gamma \vdash T_1 \;\in\; \star \qquad \Gamma \vdash T_2 \;\in\; \star}{\Gamma \vdash T_1 \to T_2 \;\in\; \star} \qquad \text{(K-A\textsc{rrow})}$$

$$\frac{\Gamma, A{\leq}T_1{:}K_1 \vdash T_2 \;\in\; \star}{\Gamma \vdash All(A{\leq}T_1{:}K_1)T_2 \;\in\; \star} \qquad \text{(K-A\textsc{ll})}$$

4.5 Typing

The rules for the typing relation $\Gamma \vdash t \in T$ are the same as the ones for $F^{\omega}_{\leq}$ from Section 2.6, modulo a few extra kinding assumptions, as we changed the syntax for universal quantification and added kinding requirements to the subtyping statements. Subsumption again connects the typing system with the subtyping part. Term abstraction and application introduce and eliminate arrow-types, likewise type abstraction and type application for universally quantified types.

$$\frac{\Gamma \vdash s \in S \quad \Gamma \vdash T \in \star \quad \Gamma \vdash S \leq T}{\Gamma \vdash s \in T} \qquad \text{(T-Subsumption)}$$

$$\frac{\vdash \Gamma \; ok}{\Gamma \vdash x \in \Gamma(x)} \qquad \text{(T-Var)}$$

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

$$\frac{\Gamma \vdash f \in T_1 \to T_2 \qquad \Gamma \vdash a \in T_1}{\Gamma \vdash f\, a \in T_2} \qquad \text{(T-Arrow-E)}$$

$$\frac{\Gamma, A{\leq}T_1{:}K_1 \vdash t \in T_2}{\Gamma \vdash \mathit{fun}(A{\leq}T_1{:}K_1)t \in \mathit{All}(A{\leq}T_1{:}K_1)T_2} \qquad \text{(T-All-I)}$$

$$\frac{\begin{array}{c}\Gamma \vdash S \leq T_1 \in K_1 \\ \Gamma \vdash f \in \mathit{All}(A{\leq}T_1{:}K_1)T_2 \qquad \Gamma \vdash S \in K_1\end{array}}{\Gamma \vdash f\, S \in [S/A]T_2} \qquad \text{(T-All-E)}$$

Chapter 5

Stratifying Kinding and Subtyping

This chapter contains the proof of proof of decidability of the subtyping relation of the previous chapter. The proof proceeds partly in parallel with the corresponding one for the pure case from Chapter 3: we again address conversion on the level of kinds by a normal-form argument and the rule of transitivity by cut-elimination. New is the mutual dependence of kinding and subtyping via the polarized application rules. Finding an algorithm is thus more challenging now, since we cannot directly address kinding and subtyping in isolation.

5.1 Introduction

This chapter contains the proof of decidability of polarized subtyping. We start with a discussion of the main difficulties to obtain an algorithm.

The principal complication of the subtyping system compared to pure $F^{\omega}_{\leq}$ is the mutual dependence of subtyping and kinding statements via the rules for polarized application and abstraction on type level:

$$\begin{tikzcd}
\text{kinding} \arrow[rr, bend left, "\text{pol. application}"] & & \text{subtyping} \arrow[ll, bend left] \\
& \text{typing} \arrow[ur, "\text{subsumption}"'] &
\end{tikzcd}$$

Figure 5.1: Polarized subtyping

Just as for $F^{\omega}_{\leq}$, the key of the argument for decidability of subtyping will be a proof of subject reduction which allows to restrict the attention on types in normal form. A direct approach would mean to treat kinding and subtyping at the same time, complicating the arguments quite a bit. Especially termination of the algorithm would get considerably more involved.

Variable occurrence and stratification of the subtyping system

Instead of a direct approach, we break the mutual dependence of kinding and subtyping, yielding a *stratified* presentation of the system where, as in $F^{\omega}_{\leq}$, subtyping still depends upon kinding as the more primitive statement, but not vice versa. This amounts to finding an independent characterization of the polarity of type operators not relying on subtyping derivations.

In Section 5.3 we will present such a characterization: instead of comparing $[A_1/A]T \leq [A_2/A]T$ under appropriate subtyping assumptions for the type variables A_1 and A_2 to characterize the polarized kind of a type operator $\mathit{Fun}(A{:}K_1)T$, we directly determine its polarity by looking at the positions in which its formal parameter A occurs inside the operator's body. This will lead to a new set of statements

$$\Gamma \vdash T \; ?_A,$$

judging the occurrence of variables in types. In the bigger part of Chapter 5 we will work within this stratified presentation. This allows a divide-and-conquer approach,

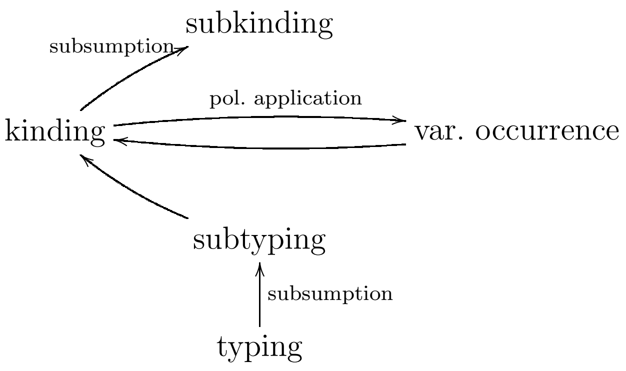

Figure 5.2: Stratifying kinding and subtyping

addressing kinding and subtyping one after the other. The dependencies between the different sets of statements are depicted in Figure 5.2.

Apart from separating the subtyping from the kinding derivations, we proceed along the lines of the proof for pure $F^{\omega}_{\leq}$ (cf. Section 3.1). As a first step towards a subtype system working exclusively on types in normal form, we distribute the effect of the conversion rule over the rest of the subtyping rules. As in the case of pure $F^{\omega}_{\leq}$, we call this directed version the *reducing system* in Section 5.6. To compensate for the effect of transitivity for type variables we again use promotion; the corresponding rule now reads:

$$\frac{S \longrightarrow^{*}_{\beta\top} S' \uparrow_{\Gamma} U \quad \Gamma \vdash S' \in K \quad \Gamma \vdash U \leq T \in K}{\Gamma \vdash S \leq T \in K} \qquad \text{(R-PROMOTE)}$$

Polarized application

A further complication over pure $F^{\omega}_{\leq}$ concerns polarized application. The corresponding rules as presented in Section 4.3 are too strict for our goal to eliminate the cut rule. For instance, assuming the upper bound of the type variable A defined as $\Gamma(A) = \mathit{Fun}(A_1{:}\star)\mathit{Fun}(A_2{:}\star)A_1 \to A_2$ and $\mathit{kind}_{\Gamma}(A) = \star \to^{-} (\star \to^{+} \star)$ the statement

$$\Gamma \vdash A\ S_1\ S_2 \leq A\ T_1\ T_2 \in \star$$

can, in general, not be derived without transitivity (assuming further $\Gamma \vdash T_1 \leq S_1 \in \star$ and $\Gamma \vdash S_2 \leq T_2 \in \star$ in accordance with the kind of A). With assistance of transitivity, a possible derivation looks as follows:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A \in \star \to^{-} (\star \to^{+} \star) \qquad \Gamma \vdash T_1 \leq S_1 \in \star}{\Gamma \vdash A\ S_1 \leq A\ T_1 \in \star \to^{+} \star}}{\Gamma \vdash A\ S_1\ S_2 \leq A\ T_1\ S_2 \in \star} \qquad \dfrac{\Gamma \vdash A\ T_1 \in \star \to^{+} \star \quad \Gamma \vdash S_2 \leq T_2 \in \star}{\Gamma \vdash A\ T_1\ S_2 \leq A\ T_1\ T_2 \in \star}}{\Gamma \vdash A\ S_1\ S_2 \leq A\ T_1\ T_2 \in \star}$$

Seemingly, we can do without transitivity here by allowing a more generous formulation of the application rules, not insisting of the two type operators in an application to coincide. In a similar way as the rule of promotion takes care of transitivity for type variables we will use more general application rules; in case of monotone operators, for example:

$$\dfrac{\begin{array}{c} S \longrightarrow^{*}_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^{*}_{\beta\top} T_1\ T_2 \\ \Gamma \vdash S_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash S_1 + \qquad \Gamma \vdash S_2 \leq T_2 \in K_1 \end{array}}{\Gamma \vdash S \leq T \in K_2}$$

allowing the two type operators S_1 and T_1 to be in subtype relation. Note that we impose a polarity requirement upon one of the operators only, in the rule above requiring nothing about the polarity of T_1. Thus T_1 may or may not be monotone and, in fact, $\Gamma \vdash S_1 \leq T_1 \in K_1 \to K_2$ implies nothing about the polarity of T_1. Of course, we thus need a symmetrical rule handling the polarity of operators on the right-hand side of a subtyping statement, as well.

5.2 Proof outline

As mentioned in the introduction , we will not carry out the subject reduction, nor the cut-elimination proof, within the original subtyping and kinding system. Instead we use a presentation where the kinding and the subtyping system are stratified in that subtyping still depends of kinding, though not vice versa, at the price of new statements $\Gamma \vdash T\ ?_A$ for the occurrence of type variables in a type. In addition, we discussed another three changes:

- reduction in the premises of the subtyping rules replaces the conversion rule,
- the rule of promotion takes care of essential uses of transitivity for type variables, and
- we use generous application rules to the same effect for applications.

As in the corresponding proof for pure $F^{\omega}_{\leq}$, our goal will be twofold: a proof that we can restrict our attention to types in normal form, thereby eliminating the non-determinism inherent in the conversion rule, and secondly elimination of the cut.

Variable occurrence

First, consider treating the non-deterministic reduction steps in the premises of the rules by always reducing the types to their unique normal form. To be able to do so amounts to a subject reduction property, i.e., preservation of subtyping under reduction. The underlying mechanism of $\beta\top$-reduction is substitution, and the key of the argument, as for pure $F^{\omega}_{\leq}$, will be preservation of subtyping under substitution. The main complication over $F^{\omega}_{\leq}$ here is that we have to deal with polarized instead of pointwise substitution. To understand the problems concerning substitution we need to have a closer look at variable occurrence (the formalization of the relation $\Gamma \vdash T\ ?_A$ follows in Section 5.3). For instance, preservation of subtyping under monotone substitution means:

> If $\Gamma \vdash S \leq T \in K$ and $\Gamma_1 \vdash U_1 \leq U_2 \in K'$ with the variable A occuring monotonely, then $\Gamma' \vdash [U_1/A]S \leq [U_1/A]T \in K$.

To start with, in the above implication it is not clear what "A occurring monotonely" means. Preservation under substitution is needed for subject reduction, so the guideline has to be the form of the application rules of the subtyping system. For the monotone case, to stick with the example, there are two corresponding rules, R-App$+_l$ and R-App$+_r$, one for each side. The above preservation property has to cover both rules by requiring A occurring monotonely inside S on the left-hand side *or* inside T on the right-hand side.

The next question is about the occurrence of the type variable A in the context. There, we have to consider in which way the subtyping system extends the contexts. Concerning possible occurrences of type variables in upper bounds, the interesting rule is the one for All-types — type operators simply add trivial *Top*-bounds to the context not containing any type variable. So, given $\Gamma \vdash All(A'{\leq}S_1{:}K_1)S_2 \leq All(A'{\leq}T_1{:}K_1)T_2 \in \star$, the rule for universally types extends the context asymmetrically, adding one upper bound as new binding to the context, where additionally we know $\Gamma \vdash S_1 \gtreqless T_1 \in K_1$. Since the last statement abbreviates two subtype derivations, $\Gamma \vdash S_1 \leq T_1 \in K_1$ and $\Gamma \vdash T_1 \leq S_1 \in K_2$, we have to require A occurring monotonely and antimonotonely at the same time, which is the same as occurring constantly. Hence the above implication more precisely reads:

> If $\Gamma_1,\ A{:}K',\ \Gamma_2 \vdash S \leq T \in K$ and $\Gamma_1 \vdash U_1 \leq U_2 \in K'$, then $\Gamma_1,\ [U_1/A]\Gamma_2 \vdash [U_1/A]S \leq [U_1/A]T \in K$, where A occurs monotonely in S or monotonely in T, and constantly in the context.

The proof of this substitution property will be an induction on the depth of inference of $\Gamma_1,\ A{:}K',\ \Gamma_2 \vdash S \leq T \in K$. Looking closer at an inductive argument and considering again, for sake of example, the monotone case with A monotone in S, but perhaps not in T, we can see two problems:

1. In case of S-Trans, how can we guarantee that the cut-type bears the same variable polarity than the relevant type (in the example, monotone polarity) in order to have the induction go through?
2. In case of S-All, asymmetrically extending the context by one of the upper bounds, how can we guarantee that the variable replaced occurs constantly in the upper bound?

The answer in both cases is, we cannot. As a matter of fact, given $\Gamma \vdash S \leq T \in K$ the polarity of the variable A in S implies nothing about the polarity of that variable in T, nor vice versa. Thus for transitivity in the first point, we have to face the fact that we know nothing about the occurrence of A in the cut-type, which is clearly not enough to have the above induction hypothesis go though for S-Trans.

The difficulty concerning universally quantified types in the second point is related. The subtyping rule for All-types, like the All-rule in full $F^{\omega}_{\leq}$, but unlike the one insisting on equal upper bounds, is asymmetric wrt. the bounds: the subgoal for $\Gamma \vdash All(A'{\leq}S_1{:}K_1)S_2 \leq All(A'{\leq}T_1{:}K_1)T_2 \in \star$ concerning the bodies of the universally quantified type reads

$$\Gamma,\ A'{\leq}S_1{:}K_1 \vdash S_2 \leq T_2 \in \star,$$

i.e., only one of the two upper bounds extends the context. It is well-known that requiring only $\Gamma \vdash T_1 \leq S_1 \in K_1$ for the upper bounds renders the subtype relation undecidable already in the second-order case of $F_{\leq}$ [Pie94]. With $\Gamma \vdash S_1 \gtrless T_1 \in K_1$, though, we are given something quite stronger here. Nevertheless, imposing a polarity requirement upon one of the two types only, suffices solely to infer that *one* of the two upper bounds contains the variable in question constantly, yet knowing nothing about the second, which may be the one used to extend the context. The additional information $\Gamma \vdash S_1 \gtrless T_1 \in K_1$ does not suffice to transfer polarity information from S_1 to T_1, nor vice versa. Responsible for this fact are the rule of promotion and the possibility of reduction: while variable polarity is preserved under reduction and under promotion, it is not guaranteed that it cannot become better by reduction or promotion.

The above analysis reveals that for subtyping it is the rule S-Trans standing in the way for a subject reduction argument, and for the mutual subtype requirement $\Gamma \vdash S \gtrless T \in K$ a used in upper bound of All-types to extend the context it is additionally the rule of promotion and the possibility of reduction. We address these two obstacles by simply circumventing them:

As in the unpolarized case, we will prove subject reduction for subtyping statements $\Gamma \vdash S \leq T \in K$ for *cut-free* derivations, i.e., in absence of S-Trans. For pairs of statements $\Gamma \vdash S \gtrless T \in K$ we additionally disallow the use of promotion, and while we are at it, throwing out reduction in the premises, as well. We capture this strong connection between two types — no reduction, no promotion, no cut — by a separate set of rules for, as we will call it, the *equivalence* of two types, written as $\Gamma \vdash S \equiv T \in K$.

The dependencies of the different levels of statements are sketched in Figure 5.3: as before, the subtyping system in its reducing variant depends on kinding as the more primitive form of statements; the same is true for equivalence, standing for a restricted form of $\Gamma \vdash S \gtrless T \in K$-statements.

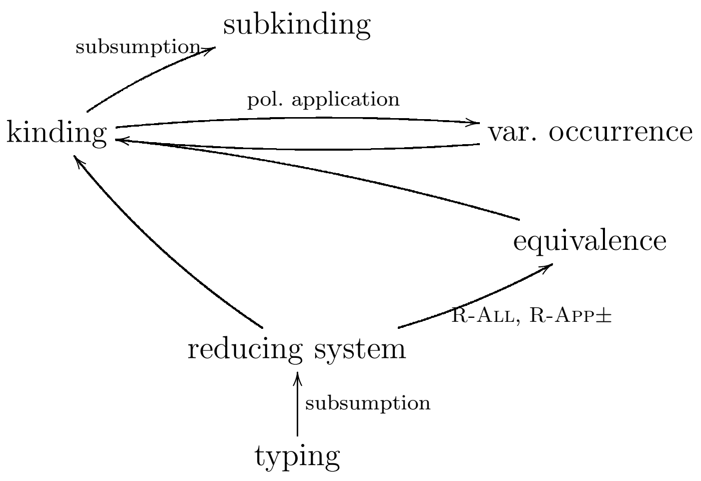

Figure 5.3: Equivalence of types

It is in the resulting restricted system in that we will carry out the subject reduction proof. Afterwards we can restrict our attention to unique normal forms forms, which we exploit for proving:

1. that transitivity is admissible using an inductive cut-elimination argument, and

2. that for an uppermost pair of statements $\Gamma \vdash S \gtrless T \in K$ with S and T already in normal form, the derivations do not contain an instance of promotion and, as a consequence, neither reduction.

Organization of the chapter

The remainder of the chapter is organized as follows. Section 5.3 introduces the independent characterization of polarized kinding based on the occurrence of type variables. The stratification allows to prove first in Section 5.4 decidability and subject reduction of the kinding system in isolation. Section 5.5 formalizes the notion of equivalence on types, used in the definition of the reducing system of Section 5.6 in which we will carry out the cut-elimination proof. Section 5.7 collects some properties of the stratified subtyping system before we prove preservation of subtyping under reduction (Section 5.8). After some digression in Section 5.9 concerning a termination measure, we proceed showing that for the proof of cut-elimination we can effectively restrict our attention to types in normal form (Section 5.10). Section 5.11 then contains a couple of technical lemmas we need to prove admissibility of cut and relaxing the restrictions of the reducing system wrt. equivalence again (Sections 5.12 and 5.13). Finally in Section 5.14 we present an algorithm for subtyping polarized $F^\omega_\leq$, proving it sound and complete.

5.3 Variable occurrence

The foremost complication compared to pure $F^\omega_\leq$ is the mutual dependence of kinding and subtyping. This section presents an alternative formulation of the kinding system — and thus indirectly of the subtyping system, as well — breaking this cycle of dependence.

Kinding

The only rules of the kinding system depending on a subtyping derivation are the ones for polarized arrow introduction. For instance, the kinding rule for monotone operators

$$\frac{\begin{array}{c}\Gamma,\, A{:}K_1 \;\vdash\; T \;\in\; K_2 \\ \Gamma,\, A_2{:}K_1,\, A_1{\leq}A_2{:}K_1 \;\vdash\; [A_1/A]T \;\leq\; [A_2/A]T \;\in\; K_2\end{array}}{\Gamma \;\vdash\; \mathit{Fun}(A{:}K_1)T \;\in\; K_1 \to^{+} K_2} \qquad \text{(K-ARROW-I+)}$$

relies on a subtyping derivation for the bodies with appropriately ordered variables substituted for the formal parameter.

To avoid this recourse to the subtyping system, we are looking for a way to determine the polarity of a kinding operator not by comparing $[A_1/A]T$ and $[A_2/A]T$ via the subtyping system, but by looking at the body T alone and determining in which way its formal parameter occurs inside T. To do so, we introduce a new set of statements $\Gamma \vdash T\ ?_A$. The polarity of A can be monotone, antimonotone, or constant. Again, we denote absence of information by the symbol $\pm$.

With these statements for variable occurrence we can write the rule for arrow introduction as follows:

$$\frac{\Gamma, A{:}K_1 \vdash T\ ?_A \qquad \Gamma, A{:}K_1 \vdash T \in K_2}{\Gamma \vdash \mathit{Fun}(A{:}K_1)T \in K_1 \rightarrow^{?} K_2} \qquad \text{(K-ARROW-I?)}$$

Remains the formalization of variable occurrence.

Variable occurrence

The definition of the rules for $\Gamma \vdash T\ ?_A$ is straightforward. With the polarities ordered as given in Section 4.4, we introduce subsumption for the corresponding polarity judgements. A type variable A occurs positively in A itself; if a variable A does not occur freely inside a type T, it occurs constantly. For arrow-types we have to take into account that the subtype relation behaves contravariantly on the left-hand side of the arrow and covariantly on its right-hand side. So, for example, we expect an arrow-rule like

$$\frac{\Gamma \vdash T_1 -_A \qquad \Gamma \vdash T_2 +_A}{\Gamma \vdash T_1 \rightarrow T_2 +_A}$$

To cut down on the number of rules it is convenient to introduce negation as unary operation on polarities, as given by Table 5.1.

$?$	$\neg ?$
$\circ$	$\circ$
$+$	$-$
$-$	$+$
$\pm$	$\pm$

Table 5.1: Negation

The polarity of a variable in an arrow-type $T_1 \rightarrow T_2$ cannot be better than either its polarity in T_2 or the negation of its polarity on the contravariant side T_1. Thus we can write the rule for arrow-types as follows:

$$\frac{\Gamma \vdash T_1\ ?^1{}_A \qquad \Gamma \vdash T_2\ ?^2{}_A \qquad ? = \neg ?^1 \vee ?^2}{\Gamma \vdash T_1 \rightarrow T_2\ ?_A}$$

For a type variable occurring with non-trivial polarity, i.e. other than $\pm$, inside an All-type, we require that it shows up constantly in the All-type's upper bound. It will turn out that requiring A to be constant in the upper bound corresponds to the requirement of the subtyping system where the upper bounds of two All-types in subtype relation have to be each one mutually smaller than the other. The occurrence of a variable inside a type operator is determined by its occurrence inside the operator's body.

Finally the statements for type applications: the polarity of A in an application $S\ T$ depends on three parts: the occurrence of A inside S, its occurrence in T, and finally the kind of S as type operator. So, for example, $\Gamma \vdash S\ T\ -_A$, if $\Gamma \vdash S\ -_A$, if S is an antimonotone operator ($\Gamma \vdash T\ -$), and $\Gamma \vdash T\ +_A$.

As in the case of arrow-types where we introduced negation, we are looking for a functional dependence between the polarity of A in S and T and the polarity of S as type operator on the one hand, and the occurrence of A in the application $S\ T$ on the other. Since $\Gamma \vdash S\ T\ ?_A$ cannot be better than $\Gamma \vdash S\ ?^1{}_A$, we can state $? = ?^1 \vee ?^2 \times ?^3$, where the commutative, binary operator $\times$ is given by Table 5.2.

$\times$	$\circ$	$+$	$-$	$\pm$
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$
$+$	$\circ$	$+$	$-$	$\pm$
$-$	$\circ$	$-$	$+$	$\pm$
$\pm$	$\circ$	$\pm$	$\pm$	$\pm$

Table 5.2: Operator $\times$ on polarities

Note that all operations introduced for polarities, including negation, are monotone in their arguments.

The relation $\Gamma \vdash T\ ?_A$ is then given inductively by the following set of rules:

$$\frac{?' \leq ? \qquad \Gamma \vdash T\ ?'_A}{\Gamma \vdash T\ ?_A}$$

$$\frac{\vdash \Gamma\ ok}{\Gamma \vdash A\ +_A}$$

$$\frac{A \notin fv(T) \qquad \vdash \Gamma\ ok}{\Gamma \vdash T\ \circ_A}$$

$$\frac{\Gamma \vdash T_1\ ?^1{}_A \qquad \Gamma \vdash T_2\ ?^2{}_A \qquad ? = \neg ?^1 \vee ?^2}{\Gamma \vdash T_1 \to T_2\ ?_A}$$

$$\frac{\Gamma \vdash T_1\ \circ_A \qquad \Gamma, A'{\le}T_1{:}K \vdash T_2\ ?\,_A \qquad A' \neq A}{\Gamma \vdash \mathit{All}(A'{\le}T_1{:}K)T_2\ ?\,_A}$$

$$\frac{\Gamma, A'{:}K \vdash T\ ?_A \qquad A' \neq A}{\Gamma \vdash \mathit{Fun}(A'{:}K)T\ ?_A}$$

$$\frac{\Gamma \vdash S\ ?^1{}_A \qquad \Gamma \vdash S\ ?^2 \qquad \Gamma \vdash T\ ?^3{}_A \qquad ? = ?^1 \vee (?^2 \times ?^3)}{\Gamma \vdash S\ T\ ?_A}$$

Fisher [Fis96] defines a similar set of operations to determine the variance (or polarity) of a type variable in a "row", a restricted form of type operator. The functions *Merge* and inversion used there correspond to our least upper bound and to negation. The calculation of polarity for applications is simpler in [Fis96], because only operators of kind $\star \to \star$ (called row functions) are treated. Hence there is no need to distinguish between constant appearance and the fact, that a variable does not occur at all.

5.4 Kinding

The aim of this section is to develop a kinding algorithm, showing decidability of kinding. The second main point is subject reduction for kinding, which we will need at different places in the coming proofs for the subtyping system. Compared to the kinding system of $F^{\omega}_{\leq}$, the task is more complicated because of the new statements for variable occurrence and since the type system no longer enjoys uniqueness of kinding.

5.4.1 Properties of the kinding system

First we collect some properties of the kinding system.

Lemma 5.1 (Partial order) The relation $\leq$ on kinds is a *partial order*, i.e. reflexive, transitive, and antisymmetric.

Lemma 5.2 (Transposition and weakening) Let Γ stand for the context Γ_1, $A_2{\leq}T_2{:}K_2$, $A_1{\leq}T_1{:}K_1$, Γ_2, with $A_2 \notin \mathit{fv}(T_1)$. Let further Γ' be be a well-formed extension of Γ_1, $A_1{\leq}T_1{:}K_1$, $A_2{\leq}T_2{:}K_2$, Γ_2.

1. If $\Gamma \vdash S\ ?_A$, then $\Gamma' \vdash S\ ?_A$.
2. If $\Gamma \vdash S \in K$, then $\Gamma' \vdash S \in K$.

The system of Section 5.3 generously allows the application of subsumption any place in a derivation. This often complicates the proofs needlessly by additional cases. Thus we first show that we can restrict the subsumption rules to the axioms, i.e., to type variables and *Top*-types, and the rules dealing with variables not occurring freely.

Definition 5.3 (Alternative presentation of kinding) An alternative presentation of the kinding relation is given by the definition from Section 5.3 with the following changes: The rules K-Subsumption and the subsumption rule for polarities are removed; likewise K-TVar, K-Top, and K-Arrow-I. In return, the following rules are added:

$$\frac{\vdash \Gamma\ \mathit{ok}}{\Gamma \vdash A\ +_A} \qquad \frac{\vdash \Gamma\ \mathit{ok}}{\Gamma \vdash A\ \pm_A} \qquad \frac{A \notin \mathit{fv}(T) \qquad \vdash \Gamma\ \mathit{ok}}{\Gamma \vdash T\ ?_{A'}}$$

$$\frac{\mathit{kind}_\Gamma A \ \leq\ K \qquad \vdash\ \Gamma\ \mathit{ok}}{\Gamma\ \vdash\ A\ \in\ K} \qquad \text{(K-TVar-Sub)}$$

$$\frac{K_1' \ \leq\ K_1 \qquad \Gamma\ \vdash\ \mathit{Top}(K_2)\ \in\ K_2'}{\Gamma\ \vdash\ \mathit{Top}(K_1 \rightarrow^? K_2)\ \in\ K_1' \rightarrow^{?'} K_2'} \qquad \text{(K-Top-Sub)}$$

$$\frac{\begin{array}{c} K_1 \ \leq\ K_1' \\ \Gamma, A{:}K_1'\ \vdash\ T\ ?_A \qquad \Gamma, A{:}K_1'\ \vdash\ T\ \in\ K_2 \end{array}}{\Gamma\ \vdash\ \mathit{Fun}(A{:}K_1')T\ \in\ K_1 \rightarrow^? K_2} \qquad \text{(K-Arrow-I-Sub)}$$

Lemma 5.4 (Generation for contexts)

1. If $\vdash \Gamma\ \mathit{ok}$, then:
   (a) $\Gamma = \bullet$; or
   (b) $\Gamma = \Gamma', x{:}T$ with $\vdash \Gamma'\ \mathit{ok}$ and $\Gamma' \vdash T \in \star$ as subderivation; or
   (c) $\Gamma = \Gamma', A{\leq}T{:}K$ with $\vdash \Gamma'\ \mathit{ok}$ and $\Gamma' \vdash T \in K$ as subderivations.
2. If $\Gamma \vdash T \in K$, then $\vdash \Gamma\ \mathit{ok}$ as subderivation.

3. If $\Gamma \vdash T\ ?_A$, then $\vdash \Gamma\ ok$ as subderivation.

(This generation property holds for the system of Section 5.3 as well as the one of Definition 5.3).

That this indeed is only a variant of the definition of Section 5.3 is expressed by the following three lemmas:

Lemma 5.5 In the system of Definition 5.3 the following holds:

1. Assume T well-kinded in Γ. If $\Gamma \vdash T\ ?_A$ and $? \leq ?'$, then $\Gamma \vdash T\ ?'_A$.
2. If $\Gamma \vdash T \in K$ and $K \leq K'$, then $\Gamma \vdash T \in K'$.

Corollary 5.6 If T is well-kinded in Γ, then $\Gamma \vdash T\ \pm_A$ for all type variables A.

Lemma 5.7 The system of Definition 5.3, containing statements $\vdash \Gamma\ ok$, $\Gamma \vdash T\ ?_A$, and $\Gamma \vdash T \in K$, is equivalent to the original definition from Section 5.3.

From now on, we will work with the more restrictive formulation of Definition 5.3, for example in the following generation lemma.

Lemma 5.8 (Generation for kinds)

1. (a) If $\Gamma \vdash S\ T\ ?_A$, then $\Gamma \vdash S\ ?'_A$ with $?' \leq ?$.
   (b) – If $\Gamma \vdash S\ T \circ_A$ and $\Gamma \not\vdash S\ \circ$, then $\Gamma \vdash T \circ_A$.
   – If $\Gamma \vdash S\ T\ ?_A$ with $\Gamma \vdash S\ +$ and $\Gamma \not\vdash S\ \circ$, then $\Gamma \vdash T\ ?'_A$ for some polarity $?'$ with $?' \leq ?$.
   – If $\Gamma \vdash S\ T\ ?_A$ with $\Gamma \vdash S\ -$ and $\Gamma \not\vdash S\ \circ$, then $\Gamma \vdash T\ ?'_A$ for some polarity $?'$ with $?' \leq \neg\, ?$.
   – If $\Gamma \vdash S\ T\ ?_A$ with $\Gamma \not\vdash S\ +$ and $\Gamma \not\vdash S\ -$, where $? \neq \pm$, then $\Gamma \vdash T \circ_A$.
   (c) If $\Gamma \vdash \mathit{Fun}(A{:}K)T\ ?_A$, then $\Gamma, A{:}K \vdash T\ ?_A$.
   (d) If $\Gamma \vdash T_1 \rightarrow T_2\ ?_A$, then $\Gamma \vdash T_1\ ?^1{}_A$ and $\Gamma \vdash T_2\ ?^2{}_A$ with two polarities $?^1$ and $?^2$ such that $? = \neg\, ?^1 \vee ?^2$.
   (e) If $\Gamma \vdash \mathit{All}(A'{\leq}T_1{:}K_1)T_2\ ?_A$ with $? \in \{\circ, +, -\}$, then $\Gamma, A'{\leq}T_1{:}K_1 \vdash T_2\ ?_A$ and $\Gamma \vdash T_1 \circ_A$.
2. (a) If $\Gamma \vdash A \in K$, then $\mathit{kind}_\Gamma A \leq K$.
   (b) If $\Gamma \vdash \mathit{Top}(\star) \in K$, then $K = \star$.
   (c) If $\Gamma \vdash \mathit{Top}(K_1 \rightarrow^? K_2) \in K$, then $K = K'_1 \rightarrow^{?'} K'_2$ with $K'_1 \leq K_1$ and $\Gamma \vdash \mathit{Top}(K_2) \in K'_2$.

(d) If $\Gamma \vdash \mathit{Fun}(A{:}K_1)T \in K$, then there are two kinds K_1' and K_2' and a polarity ? with $K = K_1' \to^? K_2'$ and $K_1' \leq K_1$. Furthermore $\Gamma, A{:}K_1 \vdash T \in K_2'$ and $\Gamma, A{:}K_1 \vdash T\ ?_A$.

(e) If $\Gamma \vdash S\ T \in K$, then there is a kinds K' and a polarity ? with $\Gamma \vdash S \in K' \to^? K$ and $\Gamma \vdash T \in K'$.

(f) If $\Gamma \vdash S \to T \in K$, then $K = \star$ where $\Gamma \vdash S \in \star$ and $\Gamma \vdash T \in \star$.

(g) If $\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \in K$, then $K = \star$ and $\Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \in \star$.

Moreover, the derivations in the "then"-parts are subderivations of the original derivations.

We have defined the operations $\vee$ and $\wedge$ as supremum and infimum on the domain of polarities. We can lift this small lattice to the level of kinds, relating all kinds that coincide after erasing the polarity information.

Definition 5.9 (Erasure) The *erasure* of a kind K, written $\mathit{erase}(K)$, is inductively defined as $\star$ for $K = \star$ and $\mathit{erase}(K_1) \to \mathit{erase}(K_2)$ for $K = K_1 \to^? K_2$.

Lemma 5.10 If $K_1 \leq K_2$, then $\mathit{erase}(K_1) = \mathit{erase}(K_2)$.

Lemma 5.11 (Erasure and kinds) If $\Gamma \vdash T \in K_1$ and $\Gamma \vdash T \in K_2$, then $\mathit{erase}(K_1) = \mathit{erase}(K_2)$

Lemma 5.12 (Lattice of kinds) Given kind K, the kinds K_i with $\mathit{erase}(K_i) = K$ and ordered by $\leq$ form a lattice.

The following lemma states that the set of kinds for a given kind and a context is closed under infimum and supremum. This is not a consequence of the previous lemma but a property of the kinding system.

Lemma 5.13 (Closure under $\wedge$ and $\vee$)

1. Assume the type T well-kinded in Γ. If $\Gamma \vdash T\ ?^1{}_A$ and $\Gamma \vdash T\ ?^2{}_A$, then $\Gamma \vdash T\ ?_A$, for $? = ?^1 \wedge ?^2$ and for $? = ?^1 \vee ?^2$.

2. If $\Gamma \vdash T \in K_1$ and $\Gamma \vdash T \in K_2$, then $\Gamma \vdash T \in K_1 \wedge K_2$ and $\Gamma \vdash T \in K_1 \vee K_2$.

Thus we get as an immediate corollary:

Corollary 5.14 If $\Gamma \vdash T\ +$ and $\Gamma \vdash T\ -$, then $\Gamma \vdash T\ \circ$.

5.4.2 Kinding algorithm

Even after the restriction of the subsumption rule, the kinding system is not syntax directed. It allows to derive different statements for type variables; the same is true for the maximal type. An algorithm for kinding should not allow such liberty, but will instead give back the *minimal* kind of the given type. We start with the corresponding definition.

Definition 5.15 (Minimal kinds) A kind K is called *minimal* for a type T in context Γ, if $\Gamma \vdash T \in K$, and for all kinds K' with $\Gamma \vdash T \in K'$, we have $K \leq K'$.

In principle, the existence of a minimal kind for a given type is already a consequence of the closure under infimum for kinds of a given type (Lemma 5.13). Moreover, it is easy to see that each kind can only bear a finite number of kinds. Hence the existence of a binary lower bound on kinds entails the existence of a minimal kind for a given type. It is, though, algorithmically not too clever to determine the minimal kind of a given type by calculating the set of possible kinds and looking for the minimal kind in it. The aim must be to find a deterministic procedure directly yielding the minimal kind. The idea of the algorithm is simple: take the rules of the kinding system (without the rule of subsumption) and give back the kind as careful, i.e., as small, as possible.

Definition 5.16 (Algorithmic kinding) The *algorithmic* relations $\Gamma \vdash_{\mathcal{A}} T \in K$ and $\Gamma \vdash_{\mathcal{A}} T\ ?_A$ are the smallest relations closed under the rules listed below.

$$\frac{A \notin fv(T) \qquad \vdash_{\mathcal{A}} \Gamma\ ok}{\Gamma \vdash_{\mathcal{A}} T \circ_A}$$

$$\frac{\vdash \Gamma\ ok}{\Gamma \vdash_{\mathcal{A}} A +_A}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1\ ?^1{}_A \qquad \Gamma \vdash_{\mathcal{A}} T_2\ ?^2{}_A \qquad ? = \neg ?^1 \vee ?^2}{\Gamma \vdash_{\mathcal{A}} T_1 \to T_2\ ?_A}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1 \circ_A \qquad \Gamma, A' \leq T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ ?_A \qquad A \neq A'}{\Gamma \vdash_{\mathcal{A}} All(A' \leq T_1{:}K_1)T_2\ ?_A}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1\ ?'_A \qquad ?' \neq \circ \qquad \Gamma, A' \leq T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ ?_A \qquad A \neq A'}{\Gamma \vdash_{\mathcal{A}} All(A' \leq T_1{:}K_1)T_2 \pm_A}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S\ ?^1{}_A \qquad \Gamma \vdash_{\mathcal{A}} S\ ?^2 \qquad \Gamma \vdash_{\mathcal{A}} T\ ?^3{}_A \qquad ? = ?^1 \vee (?^2 \times ?^3)}{\Gamma \vdash_{\mathcal{A}} S\ T\ ?_A}$$

$$\frac{\Gamma, A'{:}K \vdash_{\mathcal{A}} T\ ?_A \qquad A \neq A'}{\Gamma \vdash_{\mathcal{A}} \mathit{Fun}(A'{:}K)T\ ?_A}$$

$$\frac{\vdash \Gamma\ \mathit{ok}}{\Gamma \vdash_{\mathcal{A}} A \in \mathit{kind}_\Gamma A} \qquad \text{(K-TVar)}$$

$$\frac{\vdash \Gamma\ \mathit{ok}}{\Gamma \vdash_{\mathcal{A}} \mathit{Top}(\star) \in \star} \qquad \text{(K-Top}\star\text{)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} \mathit{Top}(K_2) \in K_2'}{\Gamma \vdash_{\mathcal{A}} \mathit{Top}(K_1 \to^? K_2) \in K_1 \to^\circ K_2'} \qquad \text{(K-Top)}$$

$$\frac{\Gamma, A{:}K_1 \vdash_{\mathcal{A}} T\ ?_A \qquad \Gamma, A{:}K_1 \vdash_{\mathcal{A}} T \in K_2}{\Gamma \vdash_{\mathcal{A}} \mathit{Fun}(A{:}K_1)T \in K_1 \to^? K_2} \qquad \text{(K-Arrow-I?)}$$

$$\frac{\begin{array}{c} K_1' \leq K_1 \\ \Gamma \vdash_{\mathcal{A}} S \in K_1 \to^? K_2 \qquad \Gamma \vdash_{\mathcal{A}} T \in K_1' \end{array}}{\Gamma \vdash_{\mathcal{A}} S\,T \in K_2} \qquad \text{(K-Arrow-E-A)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1 \in \star \qquad \Gamma \vdash_{\mathcal{A}} T_2 \in \star}{\Gamma \vdash_{\mathcal{A}} T_1 \to T_2 \in \star} \qquad \text{(K-Arrow)}$$

$$\frac{\Gamma, A{\leq}T_1{:}K_1 \vdash_{\mathcal{A}} T_2 \in \star}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star} \qquad \text{(K-All)}$$

Additionally, we write $\Gamma \vdash_{\mathcal{A}} T\ ?$, if $\Gamma \vdash_{\mathcal{A}} T \in K_1 \to^? K_2$ for some kinds K_1 and K_2.

The system of the previous definition is deterministic and syntax directed.[1] It is, of course, not equivalent to the non-algorithmic definition of the kinding relation: after all, the algorithm is not intended to give back all possible kinds for a given type, but only a special one: the minimal. Correctness and completeness of the algorithm means that the algorithm synthesizes the minimal kind for each well-kinded type:

Lemma 5.17 (Soundness and completeness of algorithmic kinding)

1. (a) If $\vdash_{\mathcal{A}} \Gamma\ \mathit{ok}$, then $\vdash \Gamma\ \mathit{ok}$.
   (b) If $\Gamma \vdash_{\mathcal{A}} T\ ?_A$, then $\Gamma \vdash T\ ?_A$.
   (c) If $\Gamma \vdash_{\mathcal{A}} T \in K$, then $\Gamma \vdash T \in K$.
2. (a) If $\vdash \Gamma\ \mathit{ok}$, then $\vdash_{\mathcal{A}} \Gamma\ \mathit{ok}$.

[1]The choice between the two rules for occurrence of variables in All-types is resolved by the occurrence of the type variable in the upper bound.

(b) If $\Gamma \vdash T\ ?_A$, then $\Gamma \vdash_{\mathcal{A}} T\ ?'_A$ with $?' \leq ?$.

(c) If $\Gamma \vdash T \in K$, then $\Gamma \vdash_{\mathcal{A}} T \in K'$ with $K' \leq K$.

As an immediate consequence we get that the algorithm gives back the minimal kind.

Corollary 5.18 (Minimality)

1. If $\Gamma \vdash_{\mathcal{A}} T\ ?_A$, then $\Gamma \vdash T\ ?_A$ is minimal.
2. If $\Gamma \vdash_{\mathcal{A}} T \in K$, then $\Gamma \vdash T \in K$ is minimal.

Finally the proof, that the system is not only deterministic, but also terminates.

Lemma 5.19 (Termination) The kinding algorithm of Definition 5.16 always terminates.

Proof: The algorithmic system of Definition 5.16, containing the statements $\vdash_{\mathcal{A}} \Gamma\ ok$, $\Gamma \vdash_{\mathcal{A}} T\ ?_A$, and $\Gamma \vdash_{\mathcal{A}} T \in K$, is syntax directed. It terminates as for each rule, proceeding from goal to the premises, the number of characters in the context and the types, but not counting the kinds on the right-hand side of the $\in$–symbol (because of K-Arrow-E-A), strictly decreases. Furthermore, the relation $\leq$ on kinds is decidable. □

Note how much more complicated a termination argument for kinding would be if we used the kinding system of Section 4.4, where the kinding statements depend on subtyping, and vice versa. Termination for an algorithm for subtyping will be a major problem, even after stratifying the two systems.

Corollary 5.20 (Decidability of the stratified kinding relation) The three relations $\vdash \Gamma\ ok$, $\Gamma \vdash S\ ?_A$, and $\Gamma \vdash S \in K$ of Section 5.3 are decidable.

Proof: By soundness and completeness of the algorithm (Lemma 5.17) and termination (Lemma 5.19). To decide $\Gamma \vdash T\ ?_A$ respectively $\Gamma \vdash T \in K$, we can use the algorithm to determine the minimal statement $\Gamma \vdash_{\mathcal{A}} T\ ?'_A$, respectively $\Gamma \vdash T \in K'$, and check $?' \leq ?$, respectively $K' \leq K$. These checks are decidable, as well. □

5.4.3 Subject reduction for kinding

In this section we will prove subject reduction, i.e., preservation under reduction, for kinding statements and variable occurrence (Lemma 5.25). With substitution as underlying mechanism of reduction, we will first need preservation of kinding under substitution. We start with some technical lemmas.

Lemma 5.21 (Generation of minimal kinds) Assume $\Gamma \vdash S\ T \in K$ and $\Gamma \vdash S\ T\ ?^2{}_A$. Assume further $\Gamma \vdash_{\mathcal{A}} S\ ?^1$ as minimal statement.

1. If $?^2 \in \{+, -, \circ\}$ and $?^1 = \pm$, then $\Gamma \vdash T\ \circ_A$; if $?^2 = \pm$, then $\Gamma \vdash T\ \pm_A$.
2. If $?^1 = +$, then $\Gamma \vdash T\ ?^2{}_A$.
3. If $?^1 = -$, then $\Gamma \vdash T\ \neg\, ?^2{}_A$.
4. If $?^1 = \circ$, then $\Gamma \vdash T\ \pm_A$.

Lemma 5.22 The operations $\vee$ and $\times$ on polarities are associative and commutative. Additionally, the following equations hold (we assume $\neg$ to bind stronger than the binary operators):

$$\begin{array}{rcl} \neg(p \vee q) & = & \neg p \vee \neg q \\ \neg(p \times q) & = & \neg p \times q \\ (p \vee q) \times r & = & (p \times r) \vee (q \times r). \end{array}$$

An important step towards subject reduction is the following substitution lemma: polarity and kinds are preserved by substitution. We will use the lemma also at different places for the preservation of the subtype relation under substitution (Lemma 5.57 and Lemma 5.58). First we need a weakening lemma for algorithmic kinding, analogous to Lemma 5.2.

Lemma 5.23 (Transposition and weakening) Let Γ abbreviate the context Γ_1, $A_2{\leq}T_2{:}K_2$, $A_1{\leq}T_1{:}$
with $A_2 \notin fv(T_1)$. Let further Γ' be be a well-formed extension of Γ_1, $A_1{\leq}T_1{:}K_1$, $A_2{\leq}T_2{:}K_2$, Γ_2.

1. If $\Gamma \vdash_{\mathcal{A}} S\ ?_A$, then $\Gamma' \vdash_{\mathcal{A}} S\ ?_A$.
2. If $\Gamma \vdash_{\mathcal{A}} S \in K$, then $\Gamma' \vdash_{\mathcal{A}} S \in K$.

Lemma 5.24 (Substitution preserves (minimal) kinding) Let Γ abbreviate the context Γ_1, $A'{:}K$, Γ_2 where $\Gamma_1 \vdash_{\mathcal{A}} U \in K$ and let Γ' stand for Γ_1, $[U/A']\Gamma_2$.

1. If $\vdash_{\mathcal{A}} \Gamma\ ok$, then $\vdash_{\mathcal{A}} \Gamma'\ ok$.
2. Assume $\Gamma \vdash_{\mathcal{A}} T\ p_A$, $\Gamma \vdash_{\mathcal{A}} T\ q_{A'}$, and $\Gamma_1 \vdash_{\mathcal{A}} U\ r_A$, with $A \neq A'$. Then $\Gamma' \vdash_{\mathcal{A}} [U/A']T\ s_A$, where $s = p \vee q \times r$.
3. If $\Gamma \vdash_{\mathcal{A}} T \in K'$, then $\Gamma' \vdash_{\mathcal{A}} [U/A']T \in K'$.

The same holds for the non-algorithmic kinding system of Definition 5.3, as well.

With preservation of kinding under substitution in hand, we are ready to show preservation of kinding under $\beta\top$-reduction. Since the previous lemma even proved preservation of minimal kinding under substitution, one could suspect preservation of minimal kinding under reduction. Rule K-Arrow-E-A of the kinding algorithm shows why we should be careful: the application of a type operator to its argument requires that the kind of the argument is at least as good as stated by the kind of the formal parameter of the operator. But the substitution Lemma 5.24 guarantees preservation of minimal kinding if the kind of the argument exactly matches the kind expected for the formal parameter. If the argument's kind is better, reduction may lead to a smaller kind.

Lemma 5.25 (Subject reduction for kinding)

1. If $\vdash \Gamma$ *ok* and $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\vdash \Gamma'$ *ok*.
2. Assume type S well-kinded in Γ. If $\Gamma \vdash S\ ?_A$ and $S \longrightarrow^*_{\beta\top} T$ with $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\Gamma \vdash T\ ?_A$.
3. If $\Gamma \vdash S \in K$ and $S \longrightarrow^*_{\beta\top} T$ with $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\Gamma \vdash T \in K$.[2]

We get as an immediate corollary a corresponding property for minimal kinds, as calculated by the kinding algorithm.

Corollary 5.26 (Subject reduction for minimal kinds)

1. If $\vdash_{\mathcal{A}} \Gamma$ *ok* and $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\vdash_{\mathcal{A}} \Gamma'$ *ok*.
2. If $\Gamma \vdash_{\mathcal{A}} S\ ?_A$ and $S \longrightarrow^*_{\beta\top} T$ with $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\Gamma' \vdash_{\mathcal{A}} T\ ?'_A$ with $?' \leq ?$.
3. If $\Gamma \vdash S \in K$ and $S \longrightarrow^*_{\beta\top} T$ with $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\Gamma' \vdash T \in K'$ with $K' \leq K$.

Corollary 5.27 Assume type T well-kinded in Γ. If $T \longrightarrow^*_{\beta\top} T'$ with $\Gamma \vdash T' \in K'$, then $\Gamma \vdash T \in K$ and $K' \leq K$.

5.5 Equivalence of types

This section defines the notion of equivalence on types and proves the basic facts about this relation. The relation will be used in the definition of the reducing system in Section 5.6. Later we will show that equivalence of two types S and T corresponds to a pair $S \leq T$ and $T \leq S$ of subtyping statements justified by two subtyping derivations without cut, without reduction, and without the rule of promotion.

[2]The relation $\longrightarrow^*_{\beta\top}$ on contexts is the obvious extension of $\beta\top$-relation from types to contexts.

Definition 5.28 (Equivalence) The relation $\Gamma \vdash S \equiv T \in K$ is inductively given by the set of rules below. We sometimes also write $S \equiv_{\Gamma}^{K} T$ for $\Gamma \vdash S \equiv T \in K$ and call S and T *equivalent* (in context Γ and for kind K).

$$\frac{\Gamma \vdash S \in K}{\Gamma \vdash S \equiv S \in K} \qquad \text{(E-Refl)}$$

$$\frac{\Gamma \vdash \mathit{Top}(K_1),\ \mathit{Top}(K_2) \in K}{\Gamma \vdash \mathit{Top}(K_1) \equiv \mathit{Top}(K_2) \in K} \qquad \text{(E-Top)}$$

$$\frac{\Gamma \vdash T_1 \equiv S_1 \in \star \qquad \Gamma \vdash S_2 \equiv T_2 \in \star}{\Gamma \vdash S_1 \to S_2 \equiv T_1 \to T_2 \in \star} \qquad \text{(E-Arrow)}$$

$$\frac{\Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1}{\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \equiv \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star} \qquad \text{(E-All)}$$

$$\frac{\Gamma, A{:}K_1' \vdash S' \equiv T' \in K_2 \qquad \Gamma, A{:}K_1' \vdash S', T' \ ?_A \qquad K_1 \leq K_1'}{\Gamma \vdash \mathit{Fun}(A{:}K_1')S' \equiv \mathit{Fun}(A{:}K_1')T' \in K_1 \to^{?} K_2} \qquad \text{(E-Abs)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S_1 \circ \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \to K_2 \quad \Gamma \vdash S_2, T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2} \qquad \text{(E-App}\circ\text{)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S_1 + \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \to K_2 \quad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2} \qquad \text{(E-App+)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S_1 - \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \to K_2 \quad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2} \qquad \text{(E-App}-\text{)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S_1 \pm \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \to K_2 \quad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2} \qquad \text{(E-App}\pm\text{)}$$

The rules of the system above closely resemble the ones of the stratified kinding system of Definition 5.3. The strong connection between the two systems is expressed by the following lemma.

Lemma 5.29 If $\Gamma \vdash S \equiv T \in K$, then $\Gamma \vdash S \in K$ and $\Gamma \vdash T \in K$.

Lemma 5.30 Suppose $K \leq K'$. If $\Gamma \vdash S \equiv T \in K$, then $\Gamma \vdash S \equiv T \in K'$.

We lift the notion of equivalence from types to contexts by the following definition.

Definition 5.31 (Equivalence of contexts) The *equivalence* of two contexts Γ_1 and Γ_2, denoted by $\vdash \Gamma_1 \equiv \Gamma_2$, is inductively defined by the following three rules:

$$\frac{}{\vdash \bullet \equiv \bullet}$$

$$\frac{\vdash \Gamma_1 \equiv \Gamma_2}{\vdash \Gamma_1, x{:}T \equiv \Gamma_2, x{:}T}$$

$$\frac{\vdash \Gamma_1 \equiv \Gamma_2 \qquad \Gamma_1 \vdash S \equiv T \in K}{\vdash \Gamma_1, A{\le}S{:}K \equiv \Gamma_2, A{\le}T{:}K}$$

Lemma 5.32 (Equivalence and kinding) Suppose $\vdash \Gamma$ *ok* and $\vdash \Gamma \equiv \Gamma'$. Then

1. $\vdash \Gamma'$ *ok*.
2. If $\Gamma \vdash T \in K$ and $\Gamma \vdash T \equiv T' \in K'$, then $\Gamma' \vdash T' \in K'$.
3. If $\Gamma \vdash T$? and $\Gamma \vdash T \equiv T' \in K$, then $\Gamma' \vdash T'$?.
4. If $\Gamma \vdash T\ ?_A$ and $\Gamma \vdash T \equiv T' \in K$, then $\Gamma' \vdash T'\ ?_A$.

Lemma 5.33 (Weakening for equivalence) Assume $A_2 \notin fv(T_1)$ and let the context Γ abbreviate $\Gamma_1, A_2{\le}T_2{:}K_2, A_1{\le}T_1{:}K_1, \Gamma_2$ Let further Γ' be be a well-formed extension of $\Gamma_1, A_1{\le}T_1{:}K_1, A_2{\le}T_2{:}K_2, \Gamma_2$. If $\Gamma \vdash S \equiv T \in K$, then $\Gamma' \vdash S \equiv T \in K$.

Next, we show that $\equiv$ is indeed an equivalence relation, proving it symmetric and transitive.

Lemma 5.34 (Symmetry and transitivity) Suppose S, T, and U well-kinded in Γ and $\vdash \Gamma \equiv \Gamma'$.

1. If $\Gamma \vdash S \equiv T \in K$, then $\Gamma' \vdash T \equiv S \in K$.
2. If $\Gamma \vdash S \equiv U \in K$ and $\Gamma' \vdash U \equiv T \in K$, then $\Gamma \vdash S \equiv T \in K$.

Symmetry of equivalence implies with Lemma 5.32, that also the minimal kinds of two equivalent types coincide:

Corollary 5.35 Suppose $\vdash_{\mathcal{A}} \Gamma$ *ok* and $\vdash \Gamma \equiv \Gamma'$.

1. If $\Gamma \vdash_{\mathcal{A}} T$? and $\Gamma \vdash T \equiv T' \in K$, then $\Gamma' \vdash_{\mathcal{A}} T'$?.
2. If $\Gamma \vdash_{\mathcal{A}} T\ ?_A$ and $\Gamma \vdash T \equiv T' \in K$, then $\Gamma' \vdash_{\mathcal{A}} T'\ ?_A$.

Since $\equiv^K_\Gamma$ is reflexive by definition, it is an equivalence relation on well-kinded types. The next substitution lemmas prove it a congruence.

Lemma 5.36 (Substitution) If $\Gamma_1, A{:}K', \Gamma_2 \vdash S \in K$ and $\Gamma_1 \vdash U_1 \equiv U_2 \in K'$, then $\Gamma_1, [U_1/A]\Gamma_2 \vdash [U_1/A]S \equiv [U_2/A]S \in K$.

Lemma 5.37 (Substitution) If $\Gamma_1, A{:}K', \Gamma_2 \vdash S_1 \equiv S_2 \in K$ and $\Gamma_1 \vdash U_1 \equiv U_2 \in K'$, then $\Gamma_1, [U_1/A]\Gamma_2 \vdash [U_1/A]S \equiv [U_2/A]T \in K$.

The following property shows that equivalence is preserved by substitution of a variable occurring constantly. Remember that the motivation to introduce equivalence was twofold. In the discussion of the form of the All-rule in Section 4.3, we argued that in the context of polarized application rules it is natural to relax the requirement of equal upper bounds for the All-types to the mutual subtype requirement and likewise to introduce application rules for the $\pm$-polarity, not insisting on equality of the argument, as in the decidable variants of $F^\omega_\leq$ or $F_\leq$ in an unpolarized setting. As neither identity on types nor $\beta\top$-equivalence is preserved by constant substitution, we introduced the weaker statements $\Gamma \vdash S \gtrless T \in K$ and $\pm$ as fourth polarity.

The relation $\equiv$ of this section was introduced for proof-technical reasons, to have a more disciplined version of $\gtrless$. The following lemma justifies the choice of $\equiv$ in that, unlike $=_{\beta\top}$, this relation is preserved by constant substitution. So in certain respect, the relation $\equiv$ for polarized kinding corresponds to identity for $F^\omega_\leq$-kinding without polarity information, and $\gtrless$ is the analogue to $=_{\beta\top}$. The second correspondence will be justified in a later section by a proof that for types in normal form the relations $\equiv$ and $\gtrless$ coincide.

First, though, we will prove preservation of equivalence under substitution, where the variable being substituted occurs constantly in the types and in the context. To be able to express this, we define constant occurrence of a variable within a context: it occurs constantly if it does so for all upper bounds in the context:[3]

Definition 5.38 Let Γ be a well-formed context and A a type variable. The relation $\vdash \Gamma \circ_A$ is inductively defined by the following three rules:

$$\vdash \bullet \circ_A$$

$$\frac{\vdash \Gamma \circ_A \qquad \Gamma \vdash T \circ_A}{\vdash \Gamma, A'{\leq}T{:}K \circ_A}$$

$$\frac{\vdash \Gamma \circ_A}{\vdash \Gamma, x{:}T \circ_A}$$

[3]It is straightforward to lift the definition from types to contexts also for polarities other than constant, but we will not need this.

Lemma 5.39 (Equivalence and constant substitution) Let context Γ abbreviate $\Gamma_1,\ A{:}K',\ \Gamma_2$ and Γ' stand for $\Gamma_1,\ [U_1/A]\Gamma_2$. Assume further $\vdash \Gamma \circ_A$ and $\Gamma_1 \vdash U_1 \in K'$ as well as $\Gamma_1 \vdash U_2 \in K'$. If $\Gamma \vdash S \equiv T \in K$ with $\Gamma \vdash S \circ_A$, then $\Gamma' \vdash [U_1/A]S \equiv [U_2/A]T \in K$

With these preservation lemmas, we can prove a simple combined confluence property of equivalence and $\beta\top$-reduction: If one of two equivalent types can do one $\beta\top$-step, the equivalence of the to types can be re-established by zero or one step of the second type. That the second type need not perform a step at all is a consequence of the fact, that equivalence of types disregards parts of the type appearing constantly. So if the redex of the first type occurs constantly, its contraction does not destroy equivalence with the second type.

Lemma 5.40 (Equivalence and $\beta\top$-reduction) Assume $\Gamma \vdash S \equiv T \in K$.

1. If $S \longrightarrow_{\beta\top} S'$, then there exists a type T' with $T \longrightarrow^{*}_{\beta\top} T'$ and $\Gamma \vdash S' \equiv T' \in K$:[4]

$$\begin{tikzcd}
S \arrow[r, equal, "\Gamma", "K"'] \arrow[d, "\beta\top"'] & T \arrow[d, dotted, "\beta\top"', "*"] \\
S' \arrow[r, dotted, equal, "\Gamma", "K"'] & T'.
\end{tikzcd}$$

2. $\Gamma \vdash S^! \equiv T^! \in K$.

As similar property holds for the promotion relation. A difference is, that now the right-hand side will do a proper step to re-establish equivalence, since the head-variable which is identical on both sides is guaranteed not to occur in constant position.

Lemma 5.41 (Equivalence and promotion) If $\Gamma \vdash S \equiv T \in K$ and $S \uparrow_\Gamma S'$, then there exists a type T' such that $T \uparrow_\Gamma T'$ and $\Gamma \vdash S' \equiv T' \in K$.

As final property of equivalence we will need its decidability.

Lemma 5.42 (Decidability of equivalence) The relation $\Gamma \vdash S \equiv T \in K$ on well-kinded types is decidable.

[4]We could strengthen the lemma by stating that $T \longrightarrow^{*}_{\beta\top} T'$ by zero or one step, but we will not need this.

Proof: The rules for equivalence are syntax directed with the exception of the ones for application. The choice of application rules is determined by the minimal polarity of the type operator, which is decidable by the corresponding algorithm (Lemma 5.20).

The application of equivalence rules must terminate since each premise for each equivalence rule is either a kinding statement (by Lemma 5.20 kinding is decidable) or a equivalence subgoal for syntactic subformulae of the original pair of types. □

5.6 The reducing system

Having formalized equivalence on types, we are now in the position to define the stratified variant of the subtyping system used for the cut-elimination proof.

Besides the changes discussed in the proof outline — separation of kinding and subtyping, introduction of R-Promote, using reduction in the premises of the subtyping rules instead of one conversion rule S-Conv, and allowing a more generous formulation of the application rules — the last distinguishing feature of the system below is a matter of convenience. Having almost duplicated the number of application rules by introducing a variant for the polarity of the operator on the left-hand side as well as for the one on the right-hand side, we will make our life easier in the proofs to come by cutting down on the number of application rules to choose from.

To this end, similar to the rules for equivalence, the system takes the application rule according to which one of the two applicator's *minimal* polarity is better, i.e. smaller. This means for the monotone rule, for instance, that we require $\Gamma \vdash_{\mathcal{A}} S_1 +$ instead of $\Gamma \vdash S_1 +$ (the kinding algorithm $\vdash_{\mathcal{A}}$ gives back this minimal polarity) and additionally that the minimal polarity of T_1 as type operator is not strictly smaller than monotone; if T_1 were a constant operator, the system would choose the analogous rule for constant application on the right-hand side.

As in the proof for pure $F^{\omega}_{\leq}$, we distribute the effect of conversion over all rules by allowing the types to be arbitrarily reduced in the premises. The rule of promotion (R-Promote) generalizes the variable rule S-TVar, taking care of the essential uses of transitivity implicit in S-TVar. The foremost difference between the two systems is not directly visible in the subtyping rules, but was already discussed in connection with kinding in Section 5.3: kinding does no longer depend on subtyping. In this way we were already able to prove properties such as subject reduction and decidability of kinding without reference to the subtyping system, so that in the following we can concentrate on the subtyping part, alone.

Definition 5.43 (Reducing system) The *reducing system* is given inductively by the set of rules of Table 5.3 on page 80, where we omit the four symmetric rules for polarized application. Moreover, the eight rules for polarized type application are ordered in such a way, that always the “best possible rule” is chosen.

Notation 5.44 Again, we distinguish derivations in different systems by marking the turnstile symbol: $\vdash_{\mathcal{O}}$ for the original, non-stratified system, $\vdash_{\mathcal{R}}$ for the reducing system, $\vdash_{\mathcal{S}}$ for strong derivations in the reducing system, $\vdash_{\mathcal{C}}$ for cut-free derivations in the reducing system, and $\vdash_{\mathcal{CS}}$ for strong, cut-free derivations in the reducing system.

The next three tasks are preservation of subtyping under reduction, elimination of transitivity, and a proof that equivalence on types is a restricted form of mutual subtype relationship. Before we address these tasks in turn, we continue in the following section with some properties of the reducing system.

5.7 Properties of the reducing system

In this section we collect a couple of properties of the reducing system just defined.

Lemma 5.45 (Preservation of kinding under promotion) If $\Gamma \vdash T \in K$ and $T \Uparrow_\Gamma T'$, then $\Gamma \vdash T' \in K$.

Note that minimal kinds are not preserved under promotion. The reason is similar to the one that prevents preservation of minimal kinding under reduction: the minimal kind of a type variable A in a context Γ is determined by its kinding declaration $\mathit{kind}_\Gamma A$, but nothing prevents its upper bound $\Gamma(A)$ from bearing a better kind that this.

Corollary 5.46 If $\Gamma \vdash_{\mathcal{A}} T \in K$ and $T \Uparrow_\Gamma T'$, then $\Gamma \vdash_{\mathcal{A}} T' \in K'$ with $K' \leq K$.

Likewise, variable occurrence is, in general, not preserved under promotion. For example, the type variable A may not occur free in type $A'\ T_1 \ldots T_n$, i.e., $\Gamma \vdash A'\ T_1 \ldots T_n\ \circ_A$, but if $\Gamma \not\vdash \Gamma(A')\ \circ_A$, then also $\Gamma \not\vdash \Gamma(A')\ T_1 \ldots T_n\ \circ_A$. The polarity of a variable is preserved under promotion, though, provided the upper bound of the variable promoted contains the variable in question in constant positions, only.

Lemma 5.47 Assume $\Gamma \vdash T \in K$ and $T \Uparrow_\Gamma T'$. If $\Gamma \vdash T\ ?_A$ and $\vdash \Gamma\ \circ_A$, then $\Gamma \vdash T'\ ?_A$.

Lemma 5.48 (Expansion preserves subtyping) Assume S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{R}} S' \leq T' \in K$ where $S \longrightarrow^*_{\beta\top} S'$ and $T \longrightarrow^*_{\beta\top} T'$, then $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$. The same is correspondingly true for cut-free and strong derivations.

$$\frac{\Gamma \vdash S \leq U \in K \quad \Gamma \vdash U \leq T \in K \quad \Gamma \vdash U \in K}{\Gamma \vdash S \leq T \in K} \qquad \text{(R-Trans)}$$

$$\frac{S \longrightarrow^*_{\beta\top} U \quad T \longrightarrow^*_{\beta\top} U \quad \Gamma \vdash U \in K}{\Gamma \vdash S \leq T \in K} \qquad \text{(R-Refl)}$$

$$\frac{S \longrightarrow^*_{\beta\top} U \uparrow_\Gamma U' \quad \Gamma \vdash U \in K \quad \Gamma \vdash U' \leq T \in K}{\Gamma \vdash S \leq T \in K} \qquad \text{(R-Promote)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} S' \quad T \longrightarrow^*_{\beta\top} \mathit{Top}(K') \\ \Gamma \vdash S' \in K \quad \Gamma \vdash \mathit{Top}(K') \in K \end{array}}{\Gamma \vdash S \leq T \in K} \qquad \text{(R-Top)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} S_1 \rightarrow S_2 \quad T \longrightarrow^*_{\beta\top} T_1 \rightarrow T_2 \\ \Gamma \vdash T_1 \leq S_1 \in \star \quad \Gamma \vdash S_2 \leq T_2 \in \star \end{array}}{\Gamma \vdash S \leq T \in \star} \qquad \text{(R-Arrow)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} \mathit{All}(A{\leq}S_1{:}K_1)S_2 \quad T \longrightarrow^*_{\beta\top} \mathit{All}(A{\leq}T_1{:}K_1)T_2 \\ \Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \leq T_2 \in \star \quad \Gamma \vdash S_1 \equiv T_1 \in K_1 \end{array}}{\Gamma \vdash S \leq T \in \star} \qquad \text{(R-All)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')S' \quad T \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')T' \\ K_1 \leq K_1' \quad \Gamma, A{:}K_1' \vdash S' \leq T' \in K_2 \quad \Gamma, A{:}K_1' \vdash S', T' \; ?_A \end{array}}{\Gamma \vdash S \leq T \in K_1 \rightarrow^? K_2} \qquad \text{(R-Abs)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} S_1 \; S_2 \quad T \longrightarrow^*_{\beta\top} T_1 \; T_2 \\ \Gamma \vdash_{\mathcal{A}} S_1 \circ \\ \Gamma \vdash S_1 \leq T_1 \in K_1 \rightarrow K_2 \quad \Gamma \vdash S_1, T_2 \in K_1 \end{array}}{\Gamma \vdash S \leq T \in K_2} \qquad \text{(R-App}\circ_l\text{)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} S_1 \; S_2 \quad T \longrightarrow^*_{\beta\top} T_1 \; T_2 \\ \Gamma \vdash_{\mathcal{A}} S_1 + \\ \Gamma \vdash S_1 \leq T_1 \in K_1 \rightarrow K_2 \quad \Gamma \vdash S_2 \leq T_2 \in K_1 \end{array}}{\Gamma \vdash S \leq T \in K_2} \qquad \text{(R-App}+_l\text{)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} S_1 \; S_2 \quad T \longrightarrow^*_{\beta\top} T_1 \; T_2 \\ \Gamma \vdash_{\mathcal{A}} S_1 - \\ \Gamma \vdash S_1 \leq T_1 \in K_1 \rightarrow K_2 \quad \Gamma \vdash T_2 \leq S_2 \in K_1 \end{array}}{\Gamma \vdash S \leq T \in K_2} \qquad \text{(R-App}-_l\text{)}$$

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} S_1 \; S_2 \quad T \longrightarrow^*_{\beta\top} T_1 \; T_2 \\ \Gamma \vdash_{\mathcal{A}} S_1 \pm \\ \Gamma \vdash S_1 \leq T_1 \in K_1 \rightarrow K_2 \quad \Gamma \vdash S_2 \equiv T_2 \in K_1 \end{array}}{\Gamma \vdash S \leq T \in K_2} \qquad \text{(R-App}\pm_l\text{)}$$

Table 5.3: Reducing system

Next we prove a simple invariant of the system, namely that if $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$ then for the normal forms of S and T we have $\Gamma \vdash S^! \in K$ and $\Gamma \vdash T^! \in K$. This means, the interpretation of the kind K in a subtyping statement $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$ is not exactly that, besides being in subtype relation, S and T are of kind K, but at least that their normal forms are. That we obtain the kinding relation only for S and T's normal form is a consequence of the fact that minimal kinds are not preserved under reduction.

Lemma 5.49 (Well-kinded subderivations) Suppose S and T well-kinded in Γ. Let d be a derivation of $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$ and d' a derivation of $\Gamma' \vdash_{\mathcal{R}} S' \leq T' \in K'$. If d' is a subderivation of d, then $\Gamma' \vdash S'^! \in K'$ and $\Gamma' \vdash T'^! \in K'$.

Lemma 5.50 (Weakening for subtyping) Assume $A_2 \notin \mathit{fv}(T_1)$ and let the context Γ abbreviate $\Gamma_1, A_2{\leq}T_2{:}K_2, A_1{\leq}T_1{:}K_1, \Gamma_2$.. Let further Γ' be be a well-formed extension of $\Gamma_1, A_1{\leq}T_1{:}K_1, A_2{\leq}T_2{:}K_2, \Gamma_2$. If $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$, then $\Gamma' \vdash_{\mathcal{R}} S \leq T \in K$. The same is correspondingly true for the $\vdash_{\mathcal{C}}$– and the $\vdash_{\mathcal{CS}}$–system.

Lemma 5.51 (Maximality of *Top*) Assume the types $\mathit{Top}(K)\ S_1 \ldots S_n$ and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K)\ S_1 \ldots S_n \leq T \in K'$, then $T \longrightarrow^*_{\beta\top} \mathit{Top}(K'')$ for some kind K''.

The next lemma shows that for a given subtyping statement one can weaken the kinding requirement. The property corresponds to subsumption for kinding in Lemma 5.5 and the analogous Lemma 5.30 for equivalence.

Lemma 5.52 Suppose S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$ and $K \leq K'$, then $\Gamma \vdash_{\mathcal{R}} S \leq T \in K'$. The same is true for cut-free derivations, strong derivations, and strong, cut-free derivations.

Lemma 5.53 Suppose S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$, where $\Gamma \vdash S \in K'$ and $\Gamma \vdash T \in K'$, then $\Gamma \vdash_{\mathcal{R}} S \leq T \in K'$. The same is true for cut-free derivations, strong derivations, and strong, cut-free derivations.

We can now begin to relate the subtyping system with equivalence on types.

Lemma 5.54 (Equivalence and subtyping) Let Γ be a well-formed context with $\vdash \Gamma \equiv \Gamma'$. Suppose in addition the types S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$ with $\Gamma \vdash S \equiv S' \in K$ and $\Gamma \vdash T \equiv T' \in K$, then $\Gamma' \vdash_{\mathcal{R}} S' \leq T' \in K$. The same is true for the $\vdash_{\mathcal{C}}$– and the $\vdash_{\mathcal{CS}}$–system.

We introduced $\Gamma \vdash S \equiv T \in K$ with the intention to characterize pairs of cut-free subtyping derivations without promotion and reduction. The following two lemmas justify this claim. Lemma 5.55 proves the easier direction: equivalence of two types implies the existence of two subtyping derivations of this restricted form. The reverse implication is stated in Lemma 5.56.

Lemma 5.55 If $\Gamma \vdash S \equiv T \in K$, then $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$ and $\Gamma \vdash_{\mathcal{C}} T \leq S \in K$ by two cut-free subtyping derivations without $\longrightarrow_{\beta\top}$-reduction and without the use of R-PROMOTE.

Lemma 5.56 Assume S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} S \gtrless T \in K$ by two cut-free subtyping derivations without $\beta\top$-reduction and without instances of R-PROMOTE, then $\Gamma \vdash S \equiv T \in K$.

Note that for two types S and T in normal form, $\Gamma \vdash_{\mathcal{C}} S \gtrless T \in K$ by two cut-free derivations without promotion implies $\Gamma \vdash S \equiv T \in K$, since the rule of promotion is the only rule that may destroy the normal form of the types in the statement. After having proven, that the types of a subtype statement can always be reduced to their unique normal form without loosing derivability, this observation will allow us replace $\gtrless$ by equivalence on types.

5.8 Subject reduction for subtyping

The task of this section is to show that, for cut-free derivations, $\beta\top$-reduction in types does not interfere with the subtyping judgment. This will allow to carry out the cut-elimination proof in the system with strong derivations so that we can rely on the subgoals being in unique normal form.

As in $F^{\omega}_{\leq}$, the core of the argument is the preservation of the subtype relation under substitution. Since the application rules of the subtyping system include polarized application, we have to deal with polarized substitution, i.e., replacement of a type variable with a given polarity by two types ordered appropriately. The application rules in the subtyping system under consideration pose requirements upon the polarity of *one* operator, only. This is mirrored in the coming substitution Lemma 5.58 by the fact that we have to prove preservation under monotone substitution, for example, if the substituted variable occurs positively only on one side of the subtyping statement.

That this weaker requirement is intuitively correct can be seen by reasoning the following way: we can use transitivity to prove the required subtype relation; assuming for example the variable A positive in S, but maybe not in T, with $U_1 \leq U_2$, we can prove in a first step $[U_1/A]S \leq [U_2/A]S$, exploiting the monotonicity of A in S. Afterwards $[U_2/A]S \leq [U_2/A]T$ by pointwise substitution, and finally we can use transitivity to obtain $[U_1/A]S \leq [U_2/A]T$.

After preservation of subtyping under polarized substitution we will proceed in much the same manner as for pure $F^{\omega}_{\leq}$, showing that the reduction of an outermost redex on either the left-hand or the right-hand side of a subtyping statement preserves its derivability (Lemma 5.61). As in the proof of Church-Rosser in Section A.1, we extend these properties to a proof of preservation of subtyping under arbitrary multi-step reduction by passing through an intermediate step where we show it for one-step parallel reduction.

Before we treat the substitution property in its general form in Lemma 5.58, the following simpler substitution lemma will take care of the case for reflexivity.

Lemma 5.57 (Substitution) Let Γ abbreviate the context $\Gamma_1,\ A{:}K',\ \Gamma_2$ and Γ' abbreviate $\Gamma_1,\ [T_1/A]\Gamma_2$. Suppose further $\Gamma_1 \vdash T_1 \in K'$ and $\Gamma_1 \vdash T_2 \in K'$. If $\Gamma \vdash S \in K$, then $\Gamma' \vdash_{\mathcal{C}} [T_1/A]S \leq [T_2/A]S \in K$, provided one of the following cases holds:

1. $\vdash \Gamma \circ_A$ and $\Gamma \vdash S \circ_A$.
2. $\vdash \Gamma \circ_A$ and $\Gamma \vdash S +_A$ and $\Gamma_1 \vdash_{\mathcal{C}} T_1 \leq T_2 \in K'$.
3. $\vdash \Gamma \circ_A$ and $\Gamma \vdash S -_A$ and $\Gamma_1 \vdash_{\mathcal{C}} T_2 \leq T_1 \in K'$.
4. $\Gamma_1 \vdash T_1 \equiv T_2 \in K'$.

In case 1 and 4 the stronger statement $\Gamma' \vdash [T_1/A]S \equiv [T_2/A]S \in K$ is implied.

Lemma 5.58 (Substitution preserves subtyping) Let Γ stand for the context $\Gamma_1,\ A{:}K',\ \Gamma_2$ and Γ' abbreviate $\Gamma_1,\ [V_1/A]\Gamma_2$. Suppose further $\Gamma_1 \vdash V_1 \in K'$ and $\Gamma_1 \vdash V_2 \in K'$, and S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$, then $\Gamma' \vdash_{\mathcal{C}} [V_1/A]S \leq [V_2/A]T \in K$, provided one of the following cases holds:

1. $\vdash \Gamma \circ_A$ and $\Gamma \vdash S \circ_A$. Moreover $\Gamma \vdash V_1 \in K'$ and $\Gamma \vdash V_2 \in K'$.
2. $\vdash \Gamma \circ_A$ and $\Gamma \vdash S +_A$ and $\Gamma_1 \vdash_{\mathcal{C}} V_1 \leq V_2 \in K'$.
3. $\vdash \Gamma \circ_A$ and $\Gamma \vdash S -_A$ and $\Gamma_1 \vdash_{\mathcal{C}} V_2 \leq V_1 \in K'$.
4. $\Gamma_1 \vdash V_1 \equiv V_2 \in K'$.

The same is true if one of the four conditions on the polarity of A holds for T on the right-hand side instead of for S on the left.

The following two lemmas will allow us to perform one outermost β-reduction step, resp. one $\top$-reduction step, on both sides of a subtyping statement. The lemma for the outermost β-step builds upon preservation of subtyping under substitution.

Lemma 5.59

1. Assume the types $S\ U_1$ and $(\mathit{Fun}(A{:}K_1')T)\ U_2$ well-kinded in context Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K_1')T \in K_1 \to^{?} K_2$, then $\Gamma \vdash_{\mathcal{C}} S\ U_1 \leq [U_2/A]T \in K_2$, provided one of the following cases holds:

   (a) $\Gamma \vdash \mathit{Fun}(A{:}K_1')T\ \circ$.

   (b) $\Gamma \vdash \mathit{Fun}(A{:}K_1')T\ +$ and $\Gamma \vdash_{\mathcal{C}} U_1 \leq U_2 \in K_1'$.

   (c) $\Gamma \vdash \mathit{Fun}(A{:}K_1')T\ -$ and $\Gamma \vdash_{\mathcal{C}} U_2 \leq U_1 \in K_1'$.

   (d) $\Gamma \vdash U_1 \equiv U_2 \in K_1'$.

2. Assume the types $(\mathit{Fun}(A{:}K_1')S)\ U_1$ and $T\ U_2$ well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K_1')S \leq T \in K_1 \to^{?} K_2$, then $\Gamma \vdash_{\mathcal{C}} [U_1/A]S \leq T\ U_2 \in K_2$, provided one of the following cases holds:

   (a) $\Gamma \vdash \mathit{Fun}(A{:}K_1')S\ \circ$.

   (b) $\Gamma \vdash \mathit{Fun}(A{:}K_1')S\ +$ and $\Gamma \vdash_{\mathcal{C}} U_1 \leq U_2 \in K_1'$.

   (c) $\Gamma \vdash \mathit{Fun}(A{:}K_1')S\ -$ and $\Gamma \vdash_{\mathcal{C}} U_2 \leq U_1 \in K_1'$.

   (d) $\Gamma \vdash U_1 \equiv U_2 \in K_1'$.

Lemma 5.60 Assume the types $\mathit{Top}(K')\ U$ and T well-kinded in Γ, and further $\mathit{Top}(K')\ U \longrightarrow_{\top} \mathit{Top}(K'')$. If $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K') \leq T \in K_1 \to^{?} K_2$, then $\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K'') \leq T\ U \in K_2$.

Lemma 5.61 (Outer β-step)

1. Assume the two types S and $(\mathit{Fun}(A{:}K_1')T)\ U$ well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K_1')T)\ U \in K$, then $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T \in K$.

2. Assume the two types $(\mathit{Fun}(A{:}K_1')S)\ U$ and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K_1')S)\ U \leq T \in K$, then $\Gamma \vdash_{\mathcal{C}} [U/A]S \leq T \in K$.

It is now straightforward to generalize preservation of subtyping under one outer reduction step to an arbitrary parallel reduction step. In the lemma we use parallel reduction of contexts (written $\Gamma \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \Gamma'$) as the pointwise extension of parallel reduction on types.

Lemma 5.62 (Parallel reduction preserves subtyping) Suppose the types S and T well-kinded in Γ.

1. If $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$ with $S \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} S'$ and $\Gamma \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \Gamma'$, then $\Gamma' \vdash_{\mathcal{C}} S' \leq T \in K$.

2. If $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$ with $T \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} T'$ and $\Gamma \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \Gamma'$, then $\Gamma' \vdash_{\mathcal{C}} S \leq T' \in K$.

After we can do one parallel step, the generalization to arbitrary reduction sequences is a direct corollary.

Corollary 5.63 (Reduction preserves subtypes) Assume the types S and T well-kinded in Γ with $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$. If $S \longrightarrow^*_{\beta\top} S'$ and $T \longrightarrow^*_{\beta\top} T'$, and furthermore $\Gamma \longrightarrow^*_{\beta\top} \Gamma'$, then $\Gamma' \vdash_{\mathcal{C}} S' \leq T' \in K$.

Corollary 5.64 Suppose the four types S, S', T, and T' well-kinded in Γ with $S =_{\beta\top} S'$ and $T =_{\beta\top} T'$. If $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$, then $\Gamma \vdash_{\mathcal{C}} S' \leq T' \in K$.

Now that we know that subtyping in cut-free derivations is preserved under reduction we can use this fact to prove that a cut-free derivation can be strengthened, i.e., turned into a derivation where all subtyping rules reduce their subgoals to normal forms. We can achieve this using Corollary 5.63 each time a subgoal not in normal form is generated. This can only be the case when R-Promote is used. We have to take care, however, that this process of re-normalizing comes to an end.

For $F^\omega_\leq$, this is comparatively easy: without polarized application rules, reducing the types in a subtyping statement did not increase the number of instances of R-Promote in the derivation. Since, on the other hand, promotion is the only rule able to generate for statements in normal form subgoals which are not, one can use the number of R-Promote's as terminating measure (cf. Lemma 3.29 on page 36).

This simple argument does not work here, because now that the arguments of two type applications in subtype relation need not be identical as for pointwise subtyping, the substitution of the two different arguments may well increase the number of R-Promote's needed to prove that after substitution the subtype relation still holds.

So we take as termination measure based on the maximal number on $\beta\top\Gamma$-steps of the two types under comparison. To cope with the asymmetric rule of All-types, we have to generalize this relation, considering Γ-steps only up-to equivalence of contexts. We come back to strengthening cut-free derivations in Section 5.10.

5.9 Strong normalization

This section proves strong termination of the combination of the relations $\longrightarrow_{\beta\top\Gamma}$ and $\equiv^K_\Gamma$, which we will use for the termination proof of the algorithm. The proof of strong normalization of this relation will be by contradiction, where the assumption of an infinite such sequence leads to an infinite $\beta\top\Gamma$–sequence. We additionally generalize Γ-reduction in that we will consider the context Γ up-to equivalence, only.

One complication compared to the case of pure $F^{\omega}_{\leq}$ is the rule for universal quantification, which asymmetrically extends the context. To include this into the Γ– and correspondingly $\beta\top\Gamma$–reduction, we must generalize these relations in that we consider the context Γ only up to equivalence. Before we can prove that a cut-free derivation can be strengthened by re-normalizing subtype statements in the derivation each time an instance of R-Promote generates a subgoal not in normal form, we prove strong-normalization of $\beta\top\Gamma\equiv$-reduction. We start with the definition of this more general reduction relation.

Definition 5.65 The binary family of relations $\longrightarrow_{\Gamma\equiv}$ on types is inductively defined by the same set of rules as $\longrightarrow_{\Gamma}$ from Definition A.10, with the exception of the rule for applications which is relaxed to:

$$\frac{A\ T_1 \dots T_n \uparrow_{\Gamma'} \Gamma'(A)\ T_1 \dots T_n \qquad \vdash \Gamma' \equiv \Gamma}{A\ T_1 \dots T_n \longrightarrow_{\Gamma\equiv} \Gamma'(A)\ T_1 \dots T_n}$$

We combine this relation with $\beta\top$-reduction in the same way as for $\longrightarrow_{\Gamma}$, writing $\longrightarrow_{\beta\top\Gamma\equiv}$ for the union of the two. We will often use the composite relation $\longrightarrow^{*}_{\beta\top} \longrightarrow_{\Gamma\equiv} \longrightarrow^{*}_{\beta\top}$, which we abbreviate by $\longrightarrow_{(\beta\top)^{*}\Gamma\equiv}$.

First we collect three properties of this this relation in connection with the equivalence relation on types and contexts.

Lemma 5.66 Suppose $\vdash \Gamma_1 \equiv \Gamma_2$ and $\Gamma_1 \vdash T \in K$. If $T \longrightarrow_{\Gamma_1} T'$, then there exists a type T'' with $T \longrightarrow_{\Gamma_2} T''$ and $\Gamma_1 \vdash T' \equiv T'' \in K$:

$$\begin{tikzcd}
T' \arrow[r, dotted, equal, "\Gamma_1", "K"'] & T'' \\
T. \arrow[u, "\Gamma_1"] \arrow[ur, dotted, "\Gamma_2"'] &
\end{tikzcd}$$

Lemma 5.67 (Equivalence and Γ-step) If $\Gamma \vdash S \equiv T \in K$ and $S \longrightarrow_{\Gamma} S'$, then there exists a type T' such that $T \longrightarrow_{\Gamma} T'$ and $\Gamma \vdash S' \equiv T' \in K$:

$$\begin{tikzcd}
S' \arrow[r, dotted, equal, "\Gamma", "K"'] & T' \\
S \arrow[u, "\Gamma"] \arrow[r, equal, "\Gamma", "K"'] & T. \arrow[u, dotted, "\Gamma"']
\end{tikzcd}$$

Lemma 5.68 Assume $\Gamma \vdash T \in K$. If $T \longrightarrow_{\Gamma\equiv} T'$, then there exists a type T'' such that $T \longrightarrow_{\Gamma} T''$ and $\Gamma \vdash T' \equiv T'' \in K$:

$$\begin{tikzcd}
T' \arrow[r, dotted, no head, "\Gamma", "K"'] & T'' \\
T. \arrow[u, "\Gamma\equiv"] \arrow[ur, dotted, "\Gamma"'] &
\end{tikzcd}$$

Now we can transfer the termination result of Section A to prove strong termination of $\beta\top\Gamma\equiv$–reduction. Note that $\longrightarrow_{(\beta\top)^*\Gamma}$ contains at least one $\uparrow_\Gamma$-step, whereas the $\beta\top$-part may be empty. The reason is that for the Γ-part of the relation we have a stronger diamond-property with respect to equivalence (Lemma 5.67 vs. Lemma 5.40): for two equivalent types, a $\beta\top$-step in one of the types does not mean that also the second has to perform a $\beta\top$-step to re-establish equivalence. Thus we have to rely on the Γ-part to construct a contradiction.

Lemma 5.69 Assume $\Gamma \vdash T \in K$. Then there is no infinite $\longrightarrow_{(\beta\top)^*\Gamma} \equiv^K_\Gamma$-sequence starting from T.

Proof: Assuming an infinite $\longrightarrow_{(\beta\top)^*\Gamma} \equiv^K_\Gamma$-sequence, starting from a well-kinded type T, use Lemma 5.40 to construct an infinite $\longrightarrow_{(\beta\top)^*\Gamma}$-sequence starting form T, contradicting the strong termination of $\longrightarrow_{(\beta\top)^*\Gamma}$ from Lemma A.20.

So suppose $\Gamma \vdash T \in K$ with an infinite sequence $T\ (\longrightarrow_{(\beta\top)^*\Gamma} \equiv^K_\Gamma)^\infty$ originating from T and assume the head of the sequence as

$$T \longrightarrow_{(\beta\top)^*\Gamma} T'_1 \equiv^K_\Gamma T_1 \longrightarrow_{(\beta\top)^*\Gamma} T'_2 \equiv^K_\Gamma T_2\ (\longrightarrow_{(\beta\top)^*\Gamma} \equiv^K_\Gamma)^\infty.$$

By an internal induction on the number of $\beta\top$-steps, using Lemma 5.40, transitivity of equivalence, and Lemma 5.67 for the Γ-step, there exists a type T''_1 such that the following diagram commutes:

$$\begin{tikzcd}
T''_2 \arrow[r, equal, "K", "\Gamma"'] & T'_2 \\
T'_1 \arrow[u, "\beta\top^*\Gamma"] \arrow[r, equal, "K", "\Gamma"'] & T_1. \arrow[u, "(\beta\top)^*\Gamma"']
\end{tikzcd}$$

By transitivity of equivalence (Lemma 5.34) we get $\Gamma \vdash T''_2 \equiv T_2 \in K$. Hence using the same corollary we can continue

$$\begin{tikzcd}
\vdots & & \vdots \\
T_3'' \arrow[rr, equal, "K", "\Gamma"'] & & T_3' \\
T_2'' \arrow[u, "(\beta\top)^*\Gamma"] \arrow[r, equal, "K", "\Gamma"'] & T_2' \arrow[r, equal, "K", "\Gamma"'] & T_2 \arrow[u, "(\beta\top)^*\Gamma"] \\
T_1' \arrow[u, "(\beta\top)^*\Gamma"] \arrow[r, equal, "K", "\Gamma"'] & T_1 \arrow[u, "(\beta\top)^*\Gamma"] & \\
T, \arrow[u, "(\beta\top)^*\Gamma"] & &
\end{tikzcd}$$

yielding an infinite $\longrightarrow_{\beta\top\Gamma}$-sequence starting from T. $\square$

We later will also need the following slight generalization.

Lemma 5.70 Assume $\Gamma \vdash T \in K$. Then there is no infinite $\longrightarrow_{(\beta\top)^*\Gamma\equiv}\ \equiv^K_\Gamma$-sequence starting from from T.

5.10 Strengthening

After this digression we show that cut-free derivations can be turned intro strong ones. We use the following measure, similar to the one that allowed to prove termination the algorithm for pure $F^\omega_\leq$.

Definition 5.71 (Rank) Assume the type U well-kinded in Γ. The *rank* of U in Γ is defined the pair (n, c), where n is the maximum length of all $\longrightarrow_{\beta\top\Gamma\equiv}$–reduction sequences starting from U, and c the number of characters in U. The rank of a well-kinded subtyping statement $\Gamma \vdash S \leq T \in K$ is the pairwise sum of the ranks of S and T. The ranks are ordered lexicographically.

Lemma 5.72 (Strengthening) Suppose S and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$, then $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

Proof of Lemma 5.72 on the preceding page: We are given $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$. By Corollary 5.63 and strong normalization of $\beta\top$-reduction we obtain $\Gamma \vdash_{\mathcal{C}} S^! \leq T^! \in K$. The derivation of this statement is not necessarily strong since the rule of promotion may generate subgoals not in normal form. By well-kindedness of subderivation one can normalize these again using the same corollary.

So see, that this process is finite, observe that for each subtyping rule the rank of the subtyping subgoals is strictly smaller than the rank of the goal. Especially for universally quantified types we have:

$$\frac{\begin{array}{cc} S \longrightarrow^!_{\beta\top} All(A{\leq}S_1{:}K_1)S_2 & T \longrightarrow^!_{\beta\top} All(A{\leq}T_1{:}K_1)T_2 \\ \Gamma,\, A{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \leq T_2 \in \star & \Gamma \vdash S_1 \equiv T_1 \in K_1 \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T \in \star}$$

The maximal $\longrightarrow_{\beta\top\Gamma\equiv}$–sequence starting from S is greater or equal than the sequence in the extended context $\longrightarrow_{(\beta\top)^*(\Gamma,A\leq S_1:K_1)\equiv}$ starting from S_2. This is also true for T, since $\vdash \Gamma,\, A{\leq}S_1{:}K_1 \equiv \Gamma,\, A{\leq}T_1{:}K_1$ and $\longrightarrow_{\beta\top\Gamma\equiv}$–reduction considers contexts only up to equivalence. $\square$

5.11 Characterization of strong, cut-free derivations

This section contains a couple of technical lemmas needed to prove admissibility of cut in Section 5.12 and eliminating promotion for pairs of statements $\Gamma \vdash_{\mathcal{CS}} S \gtrless T \in K$ in Section 5.13.

For the different forms of types we present a characterization of their subtypes together with a characterization of the respective subtype derivations; for example a subtype of an arrow-type can only be an arrow-type itself, or it can be promoted and reduced to an arrow-type. More complicated will be the treatment of the application rules.

It will often be the case that the subtyping derivation under consideration ends in a couple of instances of R-Promote. As in Section 3.5 for pure $F^\omega_\leq$, we use the reduction relation $\nearrow_\Gamma$ (cf. Definition 3.30) to describe the effect of such a derivation upon a type.

By definition, $\nearrow_\Gamma \subseteq \longrightarrow^*_{\beta\top\Gamma}$. Since equivalence on types is well-behaved with respect to $\beta\top$-reduction and to Γ-reduction we get the following corollary:

Corollary 5.73 Assume $\Gamma_1 \vdash S \equiv T \in K$ and $\vdash \Gamma_1 \equiv \Gamma_2$. If $S \nearrow^*_{\Gamma_1} \longrightarrow^!_{\beta\top} S'$, then there exists a type T' with $T \nearrow^*_{\Gamma_2} \longrightarrow^!_{\beta\top} T'$ with $\Gamma_1 \vdash S' \equiv T' \in K$.

The following two lemmas express easy reduction properties of this relation.

Lemma 5.74 Assume the application $S\ T$ well-kinded in Γ, with S and T in normal form. If $S \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\ S_1 \dots S_n$, then $S\ T \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\ S_1 \dots S_n\ T$.

Lemma 5.75 Suppose the types S, S', and T well-kinded in Γ. If $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$ and $S' \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} S$, then $\Gamma \vdash_{\mathcal{CS}} S' \leq T \in K'$ for some kind K'.

Before we come to the mentioned characterization of subtypes, we have a closer look at the kinds of applications in normal form. In general, the subtyping system cannot assure that, if $\Gamma \vdash S \leq T \in K$ and $\Gamma \vdash S \in K'$, then also $\Gamma \vdash T \in K'$, nor vice versa. For example, a monotone operator may well be smaller than a constant one. But in case of an application of the form $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \dots S_n \leq A\ T_1 \dots T_n \in K$, we can infer, that if the left-hand side carries a kind K', also the right-hand side does, and vice versa. This is expressed in:

Lemma 5.76 Assume the types $A\ S_1 \dots S_n$ and $A\ T_1 \dots T_n$ well-kinded in Γ. Then $\Gamma \vdash A\ S_1 \dots S_n \in K$ iff. $\Gamma \vdash A\ T_1 \dots T_n \in K$.

The next two lemmas characterize the form of subtyping derivation in the system with strong, cut-free derivation. We treat the case for applications seperately, as it is more involved than the rest.

Lemma 5.77 (Subtypes of an application) Assume the types S and T well-kinded in context Γ with $T \longrightarrow^!_{\beta\top} A\ T_1 \dots T_n$ for some $n \geq 0$. If $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$, then there are types $S_1, \dots, S_n$ such that $S \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\ S_1 \dots S_n$. Moreover, for all S_i and T_i with $0 \leq i \leq n$

$$\Gamma \vdash_{\mathcal{CS}} A\ S_1 \dots S_i \leq A\ T_1 \dots T_i \in K_i$$

for some kind K_i and where $K_n = K$ (we identify T_0 and S_0 with A to avoid some extra case for $i = 0$) and, for each $i > 0$, one of the following cases holds:

1. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \dots S_{i-1}\ \circ$ and $\Gamma \vdash_{\mathcal{A}} A\ T_1 \dots T_{i-1}\ \circ$.
2. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \dots S_{i-1}\ +$ and $\Gamma \vdash_{\mathcal{A}} A\ T_1 \dots T_{i-1}\ +$ with $\Gamma \vdash_{\mathcal{CS}} S_i \leq T_i \in K'_i$
3. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \dots S_{i-1}\ -$ and $\Gamma \vdash_{\mathcal{A}} A\ T_1 \dots T_{i-1}\ -$ with $\Gamma \vdash_{\mathcal{CS}} T_i \leq S_i \in K'_i$.
4. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \dots S_{i-1}\ \pm$ and $\Gamma \vdash_{\mathcal{A}} A\ T_1 \dots T_{i-1}\ \pm$ with $\Gamma \vdash S_i \equiv T_i \in K'_i$.

Additionally, in the cases 2 and 3, the mentioned subtype statements are justified by subderivations of $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

The next lemma is the analogue to the previous one, treating types other than application.

Lemma 5.78 (Characterization of subtypes) Assume the types S and T well-kinded in Γ with $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

1. If $T \xrightarrow{}^{!}_{\beta\top} T_1 \rightarrow T_2$, then $S \nearrow^{*}_{\Gamma} \xrightarrow{}^{!}_{\beta\top} S_1 \rightarrow S_2$ with $\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1 \in \star$ and $\Gamma \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$.

2. If $T \xrightarrow{}^{!}_{\beta\top} \mathit{All}(A{\leq}T_1{:}K_1)T_2$, then $S \nearrow^{*}_{\Gamma} \xrightarrow{}^{!}_{\beta\top} \mathit{All}(A{\leq}S_1{:}K_1)S_2$ with $\Gamma \vdash S_1 \equiv T_1 \in K_1$ and $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$.

3. If $T \xrightarrow{}^{!}_{\beta\top} \mathit{Fun}(A{:}K_1')T$, then $S \nearrow^{*}_{\Gamma} \xrightarrow{}^{!}_{\beta\top} \mathit{Fun}(A{:}K_1')S'$ with $\Gamma, A{:}K_1' \vdash_{\mathcal{CS}} S' \leq T' \in K_1 \rightarrow^{?} K_2$.

Additionally, in all three parts, the mentioned subtype statements are justified by a subderivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

Lemma 5.79 Assume the types S and $A\ U_1 \ldots U_n$ well-kinded in context Γ, and $A\ U_1 \ldots U_n$ in normal form. If $\Gamma \vdash_{\mathcal{CS}} S \leq A\ U_1 \ldots U_n \in K$ with $\Gamma \vdash S \in K'$, then also $\Gamma \vdash A\ U_1 \ldots U_n \in K'$.

The next lemma proves that for two applications with the same head variable and with their arguments ordered appropriately, we can indeed "promote" both sides of the subtype statement. This is a consequence of the strengthening lemma.

Lemma 5.80 Assume $A\ S_1 \ldots S_n$ and $A\ T_1 \ldots T_n$ well-kinded in Γ and in normal form. If $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_n \leq A\ T_1 \ldots T_n \in K$, then $\Gamma \vdash_{\mathcal{CS}} \Gamma(A)\ S_1 \ldots S_n \leq \Gamma(A)\ T_1 \ldots T_n \in K$, provided for all S_i and T_i with $1 \leq i \leq n$ one of the following four cases holds (we again identify S_0 and T_0 with A to avoid an extra case for $i = 1$):

1. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} \circ$ or $\Gamma \vdash_{\mathcal{A}} A\ T_1 \ldots T_{i-1} \circ$.

2. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} +$ or $\Gamma \vdash_{\mathcal{A}} A\ T_1 \ldots T_{i-1} +$, and $\Gamma \vdash_{\mathcal{CS}} S_i \leq T_i \in K_i$.

3. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} -$ or $\Gamma \vdash_{\mathcal{A}} A\ T_1 \ldots T_{i-1} -$, and $\Gamma \vdash_{\mathcal{CS}} T_i \leq S_i \in K_i$.

4. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} \pm$ or $\Gamma \vdash_{\mathcal{A}} A\ T_1 \ldots T_{i-1} \pm$, and $\Gamma \vdash T_i \equiv S_i \in K_i$.

We can now combine the last lemma with the characterization of subtypes from Lemma 5.78 and Lemma 5.77 to prove that a sequence of $\nearrow_{\Gamma}$-steps of a type on the right-hand side of a subtyping statement can indeed be mimicked by the type on the left-hand side. These two important lemmas — we again treat application separately — will help to solve one of the crucial cases in the proof of cut-elimination and will also be a cornerstone in the elimination proof of promotion.

Lemma 5.81 Assume S and T well-kinded in Γ with $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$ and $T \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} A\ T_1\ \ldots T_m$. Then $S \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} A\ S_1 \ldots S_m$ for some type $A\ S_1 \ldots S_m$ with $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_m \leq A\ T_1 \ldots T_m \in K$.

Moreover, for all S_i and T_i with $0 \leq i \leq m$

$$\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_i \leq A\ T_1 \ldots T_i \in K_i$$

for some kind K_i with $K_n = K$ (we identify T_0 and S_0 with A to avoid some extra case, handling $i = 0$) and, for all $i > 0$, at least one of the following cases holds:

1. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} \circ$ and $\Gamma \vdash A\ T_1 \ldots T_{i-1} \circ$.
2. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} +$ and $\Gamma \vdash A\ T_1 \ldots T_{i-1} +$ with $\Gamma \vdash_{\mathcal{CS}} S_i \leq T_i \in K_i'$
3. $\Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{i-1} -$ and $\Gamma \vdash A\ T_1 \ldots T_{i-1} -$ with $\Gamma \vdash_{\mathcal{CS}} T_i \leq S_i \in K_i'$.
4. $\Gamma \vdash S_i \equiv T_i \in K_i'$.

Additionally, in the cases 2 and 3 the mentioned subtype statements are justified by subderivations of $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

If further the sequence $T \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} A\ T_1\ \ldots T_m$ contains at least one promotion step, so does $S \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} A\ S_1 \ldots S_m$.

Lemma 5.82 Assume the types S and T well-kinded in Γ with $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

1. If $T \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} T_1 \to T_2$, then there exists a type $S_1 \to S_2$ with $S \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} S_1 \to S_2$. Moreover $\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1 \in \star$ and $\Gamma \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$.
2. If $T \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} \mathit{All}(A{\leq}T_1{:}K_1)T_2$, then there exists a type $\mathit{All}(A{\leq}S_1{:}K_1)S_2$ with $S \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} \mathit{All}(A{\leq}S_1{:}K_1)S_2$. Moreover $\Gamma \vdash S_1 \equiv T_1 \in K_1$ and $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$.
3. If $T \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} \mathit{Fun}(A{:}K_1)T$, then there exists a type $\mathit{Fun}(A{:}K_1)S$ with $S \nearrow_{\Gamma}^{*} \longrightarrow^{!}_{\beta\top} \mathit{Fun}(A{:}K_1)S$ and $\Gamma, A{:}K_1 \vdash_{\mathcal{CS}} S \leq T$.

Additionally for all cases, the mentioned subtype statements are justified by a subderivation of the original derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

5.12 Cut elimination

In this section we prove that S-TRANS is an admissible rule in the system with strong derivations, i.e., as in Section 3.5 we will show, that $\Gamma \vdash_{\mathcal{CS}} S \leq U \in K$ and $\Gamma \vdash_{\mathcal{CS}} U \leq T \in K$ implies $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$, assuming U well-kinded with K in Γ.

As before the proof will be by induction over the combined length of derivations of $\Gamma \vdash_{\mathcal{CS}} S \leq U \in K$ and $\Gamma \vdash_{\mathcal{CS}} U \leq T \in K$.

The inductive cut-elimination argument, comparing the immediate subgoals of both derivations works for all possible combinations of rules with one exception: the rule of promotion for the statement on the right. In this case, the subgoal reads $\Gamma \vdash_{\mathcal{CS}} U' \leq T \in K$ with $U \longrightarrow^{!}_{\beta\top} A\ U_1 \ldots U_n \uparrow_\Gamma U'$ and there is no possibility to make direct use of $\Gamma \vdash U' \leq T \in K$ for the inductive argument.

The solution in pure $F^{\omega}_{\leq}$ was to treat this case *without induction* (cf. Section 3.5). Instead, we directly were able to construct a derivation for the goal $\Gamma \vdash_{\mathcal{CS}} S \leq T$. That a direct construction, avoiding induction, is possible in the unpolarized case depends on two observations: first, we know that the normal form of the cut-type U which, by the definition of promotion, must be an application $A\ U'_1 \ldots U'_m$. Secondly, with pointwise subtyping only, we have precise knowledge about the subtypes of applications and the form of their derivation in the strong, cut-free system: the statement $\Gamma \vdash_{\mathcal{CS}} S \leq A\ U_1 \ldots U_m \in K$ implies that $S \nearrow^*_\Gamma \longrightarrow^{!}_{\beta\top} A\ U_1 \ldots U_m$. Thus the type S can seamlessly be promoted and reduced to $U = A\ U_1 \ldots U_m$ and one can obtain a derivation for the goal $\Gamma \vdash S \leq T \in K$ from the derivation of $\Gamma \vdash U \leq T \in K$ by an appropriate number of instances of R-Promote — and especially without induction.

In presence of the more general form of application rules, the above observation that allowed to avoid induction in the critical case of promotion, namely the knowledge that $S \nearrow^*_\Gamma \longrightarrow^{!}_{\beta\top} A\ U_1 \ldots U_m$, no longer holds. Instead, the system now only enjoys the weaker property of Lemma 5.77 that a cut-free derivation of the statement $\Gamma \vdash_{\mathcal{CS}} S \leq A\ U_1 \ldots U_m \in K$ implies

$$S \nearrow^*_\Gamma \longrightarrow^{!}_{\beta\top} A\ S_1 \ldots S_m.$$

Each of the types S_i is no longer identical to the corresponding U_i, as was the case for pointwise application, but related with U_i in accordance with the kind of the head variable A.

To treat the case of R-Promote in the last step for $\Gamma \vdash_{\mathcal{CS}} U \leq T \in K$, we can use the characterization of subtyping derivations from Section 5.11. For the inductive cut-elimination argument we no longer rely on the subtyping statements produced as *immediate* subgoals for both statements. Using Lemma 5.77 and 5.78 we can distinguish according to the structure of T's normal form.

So take again the arrow case as illustration. If $T \longrightarrow^{!}_{\beta\top} T_1 \to T_2$, we cannot be sure that the last rule applied for $\Gamma \vdash_{\mathcal{CS}} U \leq T \in \star$ was R-Arrow, but we know that there exists a derivation, which *eventually* does use R-Arrow, which is, after some some instances of R-Promote. Thus we know

$$U \nearrow^*_\Gamma \longrightarrow^{!}_{\beta\top} U_1 \to U_2 \quad \text{with} \quad \Gamma \vdash_{\mathcal{CS}} T_1 \leq U_1 \in \star \quad \text{and} \quad \Gamma \vdash_{\mathcal{CS}} U_2 \leq T_2 \in \star,$$

where the two subtyping statements are justified by subderivations of $\Gamma \vdash_{\mathcal{CS}} U \leq T \in \star$.

With this knowledge about the cut-type U, we continue with the left-hand statement $\Gamma \vdash_{\mathcal{CS}} S \leq U \in \star$. As a subtype of U, type S can mimic the sequence of promotions and reductions of U, resulting in

$$S \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} S_1 \to S_2 \quad \text{with} \quad \Gamma \vdash_{\mathcal{CS}} U_1 \leq S_1 \in \star \quad \text{and} \quad \Gamma \vdash_{\mathcal{CS}} S_2 \leq U_2 \in \star,$$

where again both subtyping statements are justified by subderivations of the original one for $\Gamma \vdash_{\mathcal{CS}} S \leq U \in \star$. This suffices to fit together the corresponding pairs of statements by induction, yielding $\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1 \in \star$ and $\Gamma \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$, and by R-Arrow $\Gamma \vdash_{\mathcal{CS}} S_1 \to S_2 \leq T_2 \to T_2 \in \star$. From there it is easy to "descend" the sequence of promotions and reductions $S \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} S_1 \to S_2$ back to the goal $\Gamma \vdash_{\mathcal{CS}} S \leq T \in \star$.

That this inductive argument also works for All-types, which asymmetrically extends the contexts, is a consequence of the properties of equivalence: Lemma 5.54 shows that equivalence behaves symmetrically with respect to contexts and derivability of subtyping statements is preserved under exchange of equivalent contexts.

Proposition 5.83 (Cut elimination) Suppose S, T, and U well-kinded in context Γ. If $\Gamma \vdash_{\mathcal{CS}} S \leq U \in K$ and $\Gamma \vdash_{\mathcal{CS}} U \leq T \in K$, then $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

5.13 Elimination of promotion

As explained in the proof outline in Section 5.2, the motivation to introduce equivalence on types was to have a restricted version of a pair of subtyping statements $\Gamma \vdash S \leq T \in K$ and $\Gamma \vdash T \leq S \in K$ (abbreviated by $\Gamma \vdash S \gtrless T \in K$) in that the two derivations neither use transitivity nor promotion, and not even reduction in the premises is allowed (cf. Lemma 5.55 and Lemma 5.56).These strong restrictions were necessary to be able to prove preservation of subtyping under reduction, and, more specifically, preservation of subtyping under polarized substitution (Lemma 5.58).

The original subtyping system from Section 4.3, though, uses a more natural formulation of equivalence of two types for the arguments in S-App± and the upper bounds of universally quantified types (even if we did not call it so): the requirement is that both types are mutually less or equal the other, irrespective of restrictions on the form of derivation justifying both statements.

It is now our obligation to show that we have not unduly simplified the system for sake of the subject reduction proof, i.e. to show that $\Gamma \vdash S \leq T \in K$ and $\Gamma \vdash T \leq S \in K$ together imply $\Gamma \vdash S \equiv T \in K$.

We do so in the $\vdash_{\mathcal{CS}}$-system, where the two derivations, albeit cut-free, may contain instances of R-Promote and thus may use $\beta\top$-reduction in the premises, even when starting from two type in normal form. Unlike the cut-elimination proof of the previous section, the proof of this property will not be by induction on the combined length of both derivations, but by induction on the sum of ranks of two statements.

The proof of this section highlights one difference between pure $F^{\omega}_{\leq}$ and the polarized version. In $F^{\omega}_{\leq}$, the pair of statements $\Gamma \vdash S \gtreqless T$ implies $S =_{\beta\top} T$ or, in case of normal forms, $S = T$. In the presence of the new polarized subtyping rules this no longer holds; assuming S and T again in normal form, we get the weaker relation $\Gamma \vdash S \equiv T \in K$, which enjoys preservation under constant substitution (cf. Lemma 5.39). The same difference between the two systems shows up in the two different subtyping rules for All-types: the "equal bounds" requirement has to be relaxed to the "mutual subtype"-requirement.

Proposition 5.84 (Elimination of promotion) If $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$ and $\Gamma \vdash_{\mathcal{CS}} T \leq S \in K$, then neither derivation contains an instance of R-Promote.

Starting with two types in normal form, promotion is the only rule that may destroy the normal form of the types in a subderivation. Hence we immediately get the following:

Corollary 5.85 If $\Gamma \vdash_{\mathcal{CS}} S \gtreqless T \in K$ with S and T in normal form, then $\Gamma \vdash S \equiv T \in K$.

5.14 Decidability of polarized subtyping

This section puts the pieces together, combining cut-elimination and the elimination of promotion into one proof that the original version of the subtyping system with kinding and subtyping mutually dependent, and the stratified version with one strong, cut-free derivations are equivalent, after all.

Two directions we have to show: soundness and completeness. We start with the completeness of the $\vdash_{\mathcal{CS}}$-system with respect to the original formulation of Section 4.3. Section 5.14.3 finally presents an algorithm for subtyping.

5.14.1 Completeness

We start with completeness of strong, cut-free derivations. The key Lemma 5.87 for the completeness result justifies the statements for variable occurrence of the reducing system by proving that these rules are complete wrt. to the original formulation. The proof hinges on the fact that for two equivalent types, there is no subtyping derivation, relating the two with an instance of promotion (Proposition 5.84).

First a technical lemma, collapsing two related type variables into one.

Lemma 5.86 Let $\Gamma = \Gamma_1,\ A_2{:}K',\ A_1{\le}A_2{:}K',\ \Gamma_2$ or $\Gamma = \Gamma_1,\ A_2{:}K',\ A_1{:}K',\ \Gamma_2$. Let further Γ' stand for $\Gamma_1,\ A{:}K',\ [A/A_1][A/A_2]\Gamma_2$. Suppose in addition S and T well-kinded in Γ.

1. If $\vdash \Gamma\ ok$ then $\vdash \Gamma'\ ok$.
2. If $\Gamma \vdash T \in K$, then $\Gamma' \vdash [A/A_1][A/A_2]T \in K$.
3. If $\Gamma \vdash T\ ?_{A'}$ with $A' \neq A_1$ and $A' \neq A_2$, then $\Gamma' \vdash T\ ?_{A'}$.
4. If $\Gamma \vdash_{\mathcal{CS}} S \le T \in K$ then $\Gamma' \vdash_{\mathcal{CS}} [A/A_1][A/A_2]S \le [A/A_1][A/A_2]T \in K$.

Lemma 5.87

1. If $\Gamma_1,\ A_2{:}K',\ A_1{\le}A_2{:}K',\ \Gamma_2 \vdash_{\mathcal{CS}} [A_1/A]T \le [A_2/A]T \in K$, then $\Gamma' \vdash T^{!}\ +_A$.
2. If $\Gamma_1,\ A_1{:}K',\ A_1{\le}A_2{:}K',\ \Gamma_2 \vdash_{\mathcal{CS}} [A_2/A]T \le [A_1/A]T \in K$, then $\Gamma' \vdash T^{!}\ -_A$.
3. If $\Gamma_1,\ A_2{:}K',\ A_1{:}K',\ \Gamma_2 \vdash_{\mathcal{CS}} [A_1/A]T \le [A_2/A]T \in K$, then $\Gamma' \vdash T^{!}\ \circ_A$.

In all three cases, let $\Gamma' = \Gamma_1,\ A{:}K',\ [A/A_1][A/A_2]\Gamma_2$ with $\vdash \Gamma'\ \circ_A$, and T be well-kinded in the given context of the subtyping statement.

Proposition 5.88 (Completeness)

1. If $\vdash_{\mathcal{O}} \Gamma\ ok$, then $\vdash \Gamma\ ok$.
2. If $\Gamma \vdash_{\mathcal{O}} T \in K$, then $\Gamma \vdash T^{!} \in K$.
3. If $\Gamma \vdash_{\mathcal{O}} S \le T \in K$, then $\Gamma \vdash_{\mathcal{CS}} S^{!} \le T^{!} \in K$.

Proof: All three parts by induction on the length of derivation.

The first part for contexts is straightforward, as the rules for both formulations coincide.

Also in the second part, most cases are solved straightforwardly by induction and preservation of kinding under reduction (Lemma 5.25) for the case of K-Arrow-E and of K-All; the difference between the two systems surfaces only in the arrow-introduction rules.

Case K-ARROW-I+:
$$\frac{\Gamma,\, A{:}K_1 \vdash_{\mathcal{O}} T \in K_2 \qquad \Gamma,\, A_2{:}K_1,\, A_1{\leq}A_2{:}K_1 \vdash_{\mathcal{O}} [A_1/A]T \leq [A_2/A]T \in K_2}{\Gamma \vdash_{\mathcal{O}} \mathit{Fun}(A{:}K_1)T \in K_1 \rightarrow^{+} K_2}$$

By induction $\Gamma,\, A{:}K_1 \vdash T^! \in K_2$ and $\Gamma,\, A_2{:}K_1,\, A_1{\leq}A_2{:}K_1 \vdash_{\mathcal{CS}} [A_1/A]T \leq [A_2/A]T \in K_2$. By Lemma 5.87 we get $\Gamma,\, A{:}K_1 \vdash T^! +_A$ and we can conclude by K-ARROW-I?:

$$\frac{\Gamma,\, A{:}K_1 \vdash T^! \in K_2 \qquad \Gamma,\, A{:}K_1 \vdash T^! +_A}{\Gamma \vdash \mathit{Fun}(A{:}K_1)T^! \in K_1 \rightarrow^{+} K_2}$$

The remaining three cases for arrow-introduction are similar, using Corollary 5.6 for the $\pm$-case.

Finally the cases for subtyping.

Case S-CONV:
$$\frac{S =_{\beta\top} T \qquad \Gamma \vdash_{\mathcal{O}} S, T \in K}{\Gamma \vdash_{\mathcal{O}} S \leq T \in K}$$

By induction $\Gamma \vdash S^! \in K$ and $\Gamma \vdash T^! \in K$, and the case follows with R-REFL and uniqueness of normal forms.

Case S-TRANS:
$$\frac{\Gamma \vdash_{\mathcal{O}} S \leq U \in K \qquad \Gamma \vdash_{\mathcal{O}} U \leq T \in K}{\Gamma \vdash_{\mathcal{O}} S \leq T \in K}$$

By induction we get $\Gamma \vdash_{\mathcal{CS}} S^! \leq U^! \in K$ and $\Gamma \vdash_{\mathcal{CS}} U^! \leq T^! \in K$. Thus by uniqueness of normal forms and cut-elimination (Proposition 5.83) $\Gamma \vdash_{\mathcal{CS}} S^! \leq T^! \in K$.

Case S-TVAR:
$$\frac{\Gamma \vdash_{\mathcal{O}} A \in K}{\Gamma \vdash_{\mathcal{O}} A \leq \Gamma(A) \in K}$$

By induction on part 1 of the lemma we get $\Gamma \vdash A \in K$. The context Γ is well-formed, and thus of the form $\Gamma = \Gamma_1,\, A{\leq}T{:}K',\, \Gamma_2$, i.e. $\Gamma(A) = T'$ and $K' \leq K$. By the corresponding generation lemmas $\Gamma_1 \vdash_{\mathcal{O}} T \in K'$ by a subderivation, hence by induction $\Gamma_1 \vdash T \in K'$, and further by subsumption and weakening $\Gamma \vdash T \in K$. Thus the case follows by R-PROMOTE, R-REFL, and preservation of kinding under reduction.

Case S-APP+:
$$\frac{\Gamma \vdash_{\mathcal{O}} S \in K_1 \rightarrow^{+} K_2 \qquad \Gamma \vdash_{\mathcal{O}} U_1 \leq U_2 \in K_1}{\Gamma \vdash_{\mathcal{O}} S\ U_1 \leq S\ U_2 \in K_2}$$

By induction we get $\Gamma \vdash S^! \in K_1 \rightarrow^{+} K_2$ and $\Gamma \vdash_{\mathcal{CS}} U_1^! \leq U_2^! \in K_1$. Using R-APP$+_l$ (or R-APP$+_r$) and reflexivity of subtyping, we can conclude:

$$\frac{\Gamma \vdash_{\mathcal{A}} S^! + \qquad \Gamma \vdash S^! \leq S^! \in K_1 \rightarrow K_2 \qquad \Gamma \vdash_{\mathcal{CS}} U_1^! \leq U_2^! \in K_1}{\Gamma \vdash_{\mathcal{C}} S^!\ U_1^! \leq S^!\ U_2^! \in K_2}$$

(In case, the minimal polarity of $S^!$ is $\Gamma \vdash_{\mathcal{A}} S^! \circ$, one of the constant application rules will do.) Note that the derivation is not necessarily strong, but with Lemma 5.72 (using additionally Lemma C.2, induction, and preservation of kinding under reduction to check the well-kindedness condition of this lemma) finally $\Gamma \vdash_{\mathcal{CS}} (S\ U_1)^! \leq (S\ U_2)^! \in K_2$.

Case S-App$\pm$:

$$\frac{\Gamma \vdash_{\mathcal{O}} S \in K_1 \to^{\pm} K_2 \qquad \Gamma \vdash_{\mathcal{O}} U_1 \gtrless U_2 \in K_1}{\Gamma \vdash_{\mathcal{O}} S\ U_1 \leq S\ U_2 \in K_2}$$

By induction $\Gamma \vdash S^! \in K_1 \to^{\pm} K_2$ and $\Gamma \vdash_{\mathcal{CS}} U_1^! \gtrless U_2^! \in K_1$. With the help of Lemma C.2, Corollary 5.85 yields $\Gamma \vdash U_1^! \equiv U_2^! \in K_1$. Thus by R-App$\pm_l$ (or R-App$\pm_l$) and reflexivity of subtyping we conclude:

$$\frac{\Gamma \vdash_{\mathcal{A}} S^! \pm \qquad \Gamma \vdash S^! \leq S^! \in K_1 \to K_2 \qquad \Gamma \vdash U_1^! \equiv U_2^! \in K_1}{\Gamma \vdash_{\mathcal{C}} S^!\ U_1^! \leq S^!\ U_2^! \in K_2}$$

Again, in case the minimal polarity of $S^!$ is better than $\pm$, the corresponding application rule gives the result, using Lemma 5.55 for the monotone and antimonotone case, and with Lemma 5.72 finally $\Gamma \vdash_{\mathcal{CS}} (S\ U_1)^! \leq (S\ U_2)^! \in K_2$. The remaining application cases are analogous.

Case S-All:

$$\frac{\Gamma \vdash_{\mathcal{O}} S_1 \gtrless T_1 \in K_1 \qquad \Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{O}} S_2 \leq T_2 \in \star}{\Gamma \vdash_{\mathcal{O}} \mathit{All}(A{\leq}S_1{:}K_1)S_2 \leq \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

By induction we get the two subtyping statements $\Gamma \vdash_{\mathcal{CS}} S_1^! \gtrless T_1^! \in K_1$ and furthermore $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2^! \leq T_2^! \in \star$. By Corollary 5.85 $\Gamma \vdash_{\mathcal{CS}} S_1^! \equiv T_1^! \in K_1$ (using again Lemma C.2 and preservation of kinding under reduction). By preservation of subtyping under reduction (Corollary 5.63) $\Gamma, A{\leq}S_1^!{:}K_1 \vdash_{\mathcal{CS}} S_2^! \leq T_2^! \in \star$, and the case follows with R-All. □

5.14.2 Soundness

The soundness proof basically will be by straightforward induction. The statements about variable occurrence in the reducing system, though, have no equivalent in the original fomulation. So we introduce statements for the polarity of variable occurrence using the same set of rules as in Definition 5.3. Of course, the difference here is that the polarity statements depend on kinding derivations in the original, unstratified system.

Definition 5.89 The relation $\Gamma \vdash_{\mathcal{O}} T\ ?_A$ is defined analogous to the corresponding relation for the stratified system in Definition 5.3.

Since for the extra statements $\Gamma \vdash_{\mathcal{O}} T\ ?_A$ we use exactly the same rules as in Definition 5.3, the same generation lemmas holds:

Observation 5.90 (Generation for (minimal) kinds) With the exception of the case for type operators (part 2(d)) the generation Lemma 5.8 is correspondingly valid for statements $\Gamma \vdash_{\mathcal{O}} T\ ?_A$. Also Lemma 5.21 holds.

The following lemma is the key step in the soundness proof. It basically proves in part 4 that the newly introduced statements $\Gamma \vdash_{\mathcal{O}} T\ ?_A$ of variable occurrence behave correctly wrt. to the polarized kinding definition of Section 4.4 which does not rely on variable occurrence. It can be seen as the reverse direction of Lemma 5.86 for the completeness proof. Interestingly, the proof of the lemma needs the completeness of the stratified system, already.

Lemma 5.91 Let Γ abbreviate the context Γ_0, $A_1{:}K_1$, Γ_2, $\ldots$, $A_n{:}K_n$, Γ_n for some $n \geq 0$. Let $\vec{A}$ stand for the vector of type variables $(A_1 \ldots A_n)$, and $\vec{A}'$ and $\vec{A}''$ for the primed versions $(A'_1 \ldots A'_n)$ and $(A''_1 \ldots A''_n)$, correspondingly. (We assume all variables involved to be distinct.) As further abbreviation we use Γ^l_i and Γ^r_i if $\Gamma = \Gamma^l_i, A_i{:}K_i, \Gamma^r_i$. Let finally Γ' be a context such that $[\vec{A}/\vec{A}''][\vec{A}/\vec{A}']\Gamma' = \Gamma$ (i.e. Γ' equals $\Gamma'^l_i, A''_i{:}K_i, A'_i{:}K_i, \Gamma'^r_i$ or $\Gamma'^l_i, A''_i{:}K_i, A'_i{\leq}A''_i{:}K_i, \Gamma'^r_i$, or A'_i and A''_i transposed.)

1. If $\vdash_{\mathcal{O}} \Gamma\ ok$, then $\vdash_{\mathcal{O}} \Gamma'\ ok$.
2. If $\Gamma \vdash_{\mathcal{O}} T \in K$ and T' a type such that $[\vec{A}/\vec{A}''][\vec{A}/\vec{A}']T' = T$, then $\Gamma' \vdash_{\mathcal{O}} T' \in K$.
3. If $\Gamma \vdash_{\mathcal{O}} T\ ?_{A'}$ and T' a type such that $[\vec{A}/\vec{A}''][\vec{A}/\vec{A}']T' = T$, then $\Gamma' \vdash_{\mathcal{O}} T'\ ?_{A'}$.
4. Assume $\Gamma \vdash_{\mathcal{O}} T \in K$ with T in normal form, and let σ_1 and σ_2 be two substitutions with domain $\{A_1, \ldots, A_n\}$ and where for all A_i, $\sigma_1(A_i) = A'_i$ or $\sigma_1(A_i) = A''_i$; likewise for σ_2. Then $\Gamma' \vdash_{\mathcal{O}} T\sigma_1 \leq T\sigma_2 \in K$ provided for all A_i one of the following cases hold:
   (a) $\Gamma \vdash T \circ_{A_i}$.
   (b) $\Gamma \vdash T +_{A_i}$, where $\sigma_1(A_i) = A'_i$ and $\sigma_2(A_i) = A''_i$, and furthermore $\Gamma' = \Gamma'^l_i, A''_i{:}K_i, A'_i{\leq}A''_i{:}K_i, \Gamma'^r_i$.
   (c) $\Gamma \vdash T -_{A_i}$, where $\sigma_1(A_i) = A'_i$ and $\sigma_2(A_i) = A''_i$, and furthermore $\Gamma' = \Gamma'^l_i, A'_i{:}K_i, A''_i{\leq}A'_i{:}K_i, \Gamma'^r_i$.
   (d) $\sigma_1(A_i) = \sigma_2(A_i)$.

   In the cases a) – c) we additionally assume $\vdash_{\mathcal{O}} \Gamma \circ_{A_i}$.

With the knowledge that variable occurrence behaves as expected, the soundness proof is straightforward.

Lemma 5.92 (Soundness)

1. If $\vdash \Gamma$ *ok* , then $\vdash_{\mathcal{O}} \Gamma$ *ok* .
2. If $\Gamma \vdash T\ ?_A$, then $\Gamma \vdash_{\mathcal{O}} T\ ?_A$.
3. If $\Gamma \vdash T \in K$, then $\Gamma \vdash_{\mathcal{O}} T \in K$.
4. If $\Gamma \vdash S \equiv T \in K$, then $\Gamma \vdash_{\mathcal{O}} S \leq T \in K$ and $\Gamma \vdash_{\mathcal{O}} T \leq S \in K$.
5. Assume S und T well-kinded in context Γ. If $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$, then $\Gamma \vdash_{\mathcal{O}} S^! \leq T^! \in K$.

5.14.3 Algorithm

We have now arrived at a reasonably deterministic system. We have stratified the system by introducing statements for variable occurrence, eliminated the cut-rule of transitivity, have shown that $\Gamma \vdash S \gtrless T \in K$ can be reduced to checking $\Gamma \vdash S \equiv T \in K$, and have proven that reducing types to normal form does not loose any expressiveness. But there are still some ambiguities concerning the rule of promotion, and concerning kinding.

Kinding

Let us first consider kinding. As it stands, the subtyping system defines a quarterny relation between two types S and T, a context, and a kind. Our goal is to have the algorithm answer the question whether S is smaller than T in the given context, determining on-the-fly a corresponding kind. As with the kinding algorithm, the natural way to proceed is to determine in each subtyping rule an appropriate kind as strict as possible and relaxing the kinding condition using the least upper bound of two kinds whenever necessary. (This strategy is justified by well-kindedness of subderivations in Lemma 5.49 and especially Lemma 5.52 that allows for a subtyping statement $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$ to weaken K to any kind K' with $K \leq K'$ without loosing derivability.) This means the algorithm does not simply yield a boolean answer whether or not S is smaller then T in a given context, but returns the minimal kind K for which $\Gamma \vdash S \leq T \in K$ in case of success, and *false*, if S and T are unrelated in the given context. In a similar vein we adapt the algorithm for checking the equivalence of two types.

Promotion vs. application

The second ambiguity mentioned above concerns the application rules on the one hand and the promotion rule on the other. In cases where we have to compare two applications $A\ S_1 \ldots S_n$ and $A'\ T_1 \ldots T_m$ and where $A \neq A'$, the situation is clear:

Lemma 5.77 tells us, even if taking one of the application rules might lead to a success, an algorithm might as well choose R-Promote.

This means we can make the system even more deterministic by allowing applications only in cases where the head variables on both sides of the subtyping statement coincide. In these cases, where $\Gamma \vdash A\ S_1 \ldots S_n \leq A\ T_1 \ldots T_n \in K$, we can additionally use the identity of the kinds on both sides (Lemma 5.76), so that the distinction between R-App+$_l$ and R-App+$_r$, for example, is no longer necessary. In this way we can simplify $\vdash_{\mathcal{CS}}$-system a little more, allowing only four application rules instead of eight, one for each polarity. In the monotone case, for instance, it suffices to have one rule

$$\frac{\begin{array}{c} S \xrightarrow{}{}^{!}_{\beta\top} A\ S_1 \ldots S_n \quad T \xrightarrow{}{}^{!}_{\beta\top} A\ T_1 \ldots T_n \quad K_1' \leq K_1 \\ \Gamma \vdash_{\mathcal{A}} S_n \leq T_n \in K_1' \\ \Gamma \vdash_{\mathcal{A}} A\ S_1 \ldots S_{n-1} \leq A\ T_1 \ldots T_{n-1} \in K_1 \to^{+} K_2 \end{array}}{\Gamma \vdash_{\mathcal{A}} S \leq T \in K_2} \quad \text{(A-App+)}$$

insisting on the head variable on both sides to coincide.

As we have just argued, we need not use one of the application rules for types whose head variables are not identical. More complicated is the question, whether or not we should use the rule of promotion in case they do.

At first glance, one might think that in a situation like $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_n \leq A\ T_1 \ldots T_n \in K$ the rule of promotion is the wrong choice. The following example shows that this is not true:

$$\frac{A\ (A\ T) \uparrow_{\Gamma} Id\ (A\ T) \xrightarrow{}{}^{!}_{\beta\top} A\ T \qquad \Gamma \vdash_{\mathcal{CS}} A\ T \leq A\ T \in \star}{\Gamma \vdash_{\mathcal{CS}} A\ (A\ T) \leq A\ T \in \star}$$

where $\Gamma = \Gamma, A{\leq}Id{:}\star \to^{+} \star$. In this example, though, there exists a derivation ending with R-App+, as well, so it might still be the case that for two applications with the same head variable, it doesn't matter whether to use R-Promote or one of the application rules, and taking an application rule would be at least a safe choice. The following example proves otherwise:

Example 5.93 Let $S_1' = Fun(A'{:}\star)A\ (S_1\ A')$ and $S_2' = Fun(A''{:}\star \to \star)A''\ S_2$ and furthermore $\Gamma = A{\leq}S_2'{:}(\star \to \star) \to^{+} \star$. We can thus derive (assuming $S_1\ S_2$ in normal form with the kinds $\Gamma \vdash S_1 \in \star \to (\star \to \star)$ and $\Gamma \vdash S_2 \in \star$):

$$\frac{A\ S_1' \uparrow_{\Gamma} S_2'\ S_1' \qquad \dfrac{S_2'\ S_1' \xrightarrow{}{}^{!}_{\beta\top} A\ (S_1\ S_2) \qquad \Gamma \vdash A\ (S_1\ S_2) \in \star}{\Gamma \vdash_{\mathcal{CS}} S_2'\ S_1' \leq A\ (S_1\ S_2) \in \star}\ \text{R-Refl}}{\Gamma \vdash_{\mathcal{CS}} A\ S_1' \leq A\ (S_1\ S_2) \in \star}\ \text{R-Promote}$$

On the other hand, this time there is no derivation ending with an instance of R-App+:

$$\frac{\Gamma \vdash_{\mathcal{CS}} A \leq A \in \star \rightarrow^{+} \star \qquad \dfrac{?}{\Gamma \vdash_{\mathcal{CS}} S_1' = \mathit{Fun}(A'{:}\star)A\ (S_1\ A') \leq A\ (S_1\ S_2) \in \star}}{\Gamma \vdash_{\mathcal{CS}} A\ S_1' \leq A\ (S_1\ S_2) \in \star}$$

For the right subgoal there is no rule applicable. The example thus shows that for statements $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_n \leq A\ T_1 \ldots T_n \in K$ there are cases where we crucially depend on R-Promote.

On the other hand it is easy to think of situations where taking the promotion rule is the wrong choice. This means that for checking an application of the form $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_n \leq A\ T_1 \ldots T_n \in K$ there is no way to know beforehand which direction to take: if the algorithm tries to use one of the application rules it has to call itself checking the subtype relation for all pairs of arguments. If the check fails, still the promotion rule might lead to a success. The effort, though, spent in checking pairs S_i and T_i is, in general, lost. And choosing the promotion rule might lead to a dead end, as well. This means that in these cases the algorithm has to *backtrack*.

Remember that with pointwise subtyping only, the dilemma is not present: the *identity* of the arguments of two applications $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_n \leq A\ T_1 \ldots T_n \in K$ (i.e. the identity of the two applications in normal form) can be "locally" checked, i.e. without resorting to a recursive call for each pair of S_i and T_i.

So, finally, we arrived at a subtyping algorithm.

Definition 5.94 (Subtyping algorithm) Assume S and T well-kinded in context Γ and in normal form. The subtyping algorithm $\Gamma \vdash_{\mathcal{A}} S \leq T \in K$ is given inductively by the rules below.

$$\frac{\Gamma \vdash_{\mathcal{A}} U \in K}{\Gamma \vdash_{\mathcal{A}} U \leq U \in K} \qquad \text{(A-Refl)}$$

$$\frac{S \uparrow_{\Gamma} \longrightarrow^{!}_{\beta\top} U \quad \Gamma \vdash_{\mathcal{A}} S \in K_1 \quad \Gamma \vdash_{\mathcal{A}} U \leq T \in K_2}{\Gamma \vdash_{\mathcal{A}} S \leq T \in K_1 \vee K_2} \qquad \text{(A-Promote)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} S \in K \qquad \Gamma \vdash_{\mathcal{A}} \mathit{Top}(K') \in K}{\Gamma \vdash_{\mathcal{A}} S \leq \mathit{Top}(K') \in K_1 \vee K_2} \qquad \text{(A-Top)}$$

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1 \leq S_1 \in \star \qquad \Gamma \vdash_{\mathcal{A}} S_2 \leq T_2 \in \star}{\Gamma \vdash_{\mathcal{A}} S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2 \in \star} \qquad \text{(A-Arrow)}$$

$$\frac{\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{A}} S_2 \leq T_2 \in \star \qquad \Gamma \vdash_{\mathcal{A}} S_1 \equiv T_1 \in K_1}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A{\leq}S_1{:}K_1)S_2 \leq \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star} \quad \text{(A-ALL)}$$

$$\frac{\begin{array}{c}\Gamma, A{:}K_1 \vdash_{\mathcal{A}} S' \,?'_A \qquad \Gamma, A{:}K_1 \vdash_{\mathcal{A}} T' \,?'_A \qquad ? = ?' \vee ?'' \\ \Gamma, A{:}K_1 \vdash_{\mathcal{A}} S' \leq T' \in K_2\end{array}}{\Gamma \vdash_{\mathcal{A}} \mathit{Fun}(A{:}K_1)S' \leq \mathit{Fun}(A{:}K_1)T' \in K_1 \to^{?} K_2} \quad \text{(A-ABS)}$$

$$\frac{\begin{array}{c}K_1' \leq K_1 \qquad \Gamma \vdash_{\mathcal{A}} S_n \equiv T_n \in K_1' \\ \Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_{n-1} \leq A\, T_1 \ldots T_{n-1} \in K_1 \to K_2\end{array}}{\Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_n \leq A\, T_1 \ldots T_n \in K_2} \quad \text{(A-APP}\pm\text{)}$$

$$\frac{\begin{array}{c}K_1' \leq K_1 \qquad \Gamma \vdash_{\mathcal{A}} S_n \leq T_n \in K_1 \\ \Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_{n-1} \leq A\, T_1 \ldots T_{n-1} \in K_1 \to^{+} K_2\end{array}}{\Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_n \leq A\, T_1 \ldots T_n \in K_2} \quad \text{(A-APP+)}$$

$$\frac{\begin{array}{c}K_1' \leq K_1 \qquad \Gamma \vdash_{\mathcal{A}} T_n \leq S_n \in K_1 \\ \Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_{n-1} \leq A\, T_1 \ldots T_{n-1} \in K_1 \to^{-} K_2\end{array}}{\Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_n \leq A\, T_1 \ldots T_n \in K_2} \quad \text{(A-APP}-\text{)}$$

$$\frac{\begin{array}{c}K_1' \leq K_1 \quad K_1'' \leq K_1 \quad \Gamma \vdash_{\mathcal{A}} S_n \in K_1' \quad \Gamma \vdash_{\mathcal{A}} T_n \in K_1'' \\ \Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_{n-1} \leq A\, T_1 \ldots T_{n-1} \in K_1 \to^{\circ} K_2\end{array}}{\Gamma \vdash_{\mathcal{A}} A\, S_1 \ldots S_n \leq A\, T_1 \ldots T_n \in K_2} \quad \text{(A-APP}\circ\text{)}$$

Proposition 5.95 (Termination of the algorithm) Suppose $\Gamma \vdash S \in K_S$ and $\Gamma \vdash T \in K_T$ and both types in normal form. Then the system of Definition 5.94 always terminates with Γ, S, and T as input.

Proof: The assumption that the rules of the algorithm can generate a tree of subtyping statements of infinite depth contradicts strong termination (Lemma 5.70) of $\longrightarrow_{\beta\mathrm{T}\Gamma\equiv}$-reduction. □

Proposition 5.96 (Decidability of subtyping) The subtyping relation $\Gamma \vdash S \leq T \in K$ is decidable.

Proof: That in order to decide whether $\Gamma \vdash S \leq T \in K$, we can use the terminating procedure of Definition 5.94, is a is a consequence of the following:

$$\Gamma \vdash_{\mathcal{O}} S \leq T \in K \text{ iff. } \Gamma \vdash_{\mathcal{A}} S^{!} \leq T^{!} \in K' \text{ for some } K' \leq K.$$

By soundness of completeness of strong, cut-free derivations (Proposition 5.92 and 5.88) we know, that the original subtyping system is equivalent to the stratified variant with strong, cut-free derivations. That we can restrict the algorithm to the four A-APP?–rules instead of the eight R-APP?$_l$– and R-APP?$_r$–rules follows from the characterization of subtypes of an applications in Lemma 5.77 and from Lemma 5.76. That finally the subtyping algorithm gives back a kind less or equal to the non-algorithmic system can be proven by straightforward induction, using the generation lemma for kinds, soundness and completeness of the kinding algorithm, well-kindedness of subderivations, and the properties of the subkinding relation. □

Chapter 6

Typing

At the end of the previous chapter, we derived an algorithm for checking the polarized subtyping relation by controlling the non-syntax-directed rules of transitivity and conversion. Now we transfer this result to the typing relation, eliminating the rule of subsumption from the system defined in Section 4.5 and accounting for its effects by extending some of the other rules.[1]

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

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

Lemma 6.1 If $\Gamma \vdash t \in T$, then $\Gamma \vdash T \in \star$.

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

Definition 6.2 (Minimal, arrow-minimal, and All-minimal types)

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

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

[1]Basically the same development as given below proves also decidability of typing for the pure case. Cf. [PS97].

3. A type $All(A{\le}S_1{:}K_1)S_2$ is *All-minimal* for s in Γ if $\Gamma \vdash s \in All(A{\le}S_1{:}K_1)S_2$ and, for all All-types $All(A{\le}T_1{:}K_1)\ T_2$ with $\Gamma \vdash s \in All(A{\le}T_1{:}K_1)\ T_2$, we have $\Gamma \vdash All(A{\le}S_1{:}K_1)\ S_2 \le All(A{\le}T_1{:}K_1)\ T_2 \in \star$.

We next show how All-minimal and arrow-minimal types of a term can be calculated from its minimal type. First we transfer Lemma 5.78 to the original subtyping system. In the following we write $S \nearrow^!_\Gamma T$ if $S \nearrow^*_\Gamma T$ and if there exists not T' such that $T \nearrow_\Gamma T'$ (Cf. Definition 3.30).

Corollary 6.3

1. Let S and $T_1 \to T_2$ be well-kinded in Γ. If $\Gamma \vdash S \le T_1 \to T_2 \in \star$, then $S \nearrow^!_\Gamma \longrightarrow^!_{\beta\top} S_1 \to S_2$ for some type $S_1 \to S_2$ with $\Gamma \vdash S_1 \to S_2 \le T_1 \to T_2 \in \star$.

2. Let S and $All(A{\le}T_1{:}K_1)T_2$ be well-kinded in Γ. If $\Gamma \vdash S \le All(A{\le}T_1{:}K_1)T_2 \in \star$, then $S \nearrow^!_\Gamma \longrightarrow^!_{\beta\top} All(A{\le}S_1{:}K_1)S_2$ for some type $All(A{\le}S_1{:}K_1)S_2$ with $\Gamma \vdash All(A{\le}S_1{:}K_1)S_2 \le All(A{\le}T_1{:}K_1)T_2 \in \star$.

Corollary 6.4

1. If the type S is minimal for s in Γ and $S \nearrow^!_\Gamma \longrightarrow^!_{\beta\top} S_1 \to S_2$, then $S_1 \to S_2$ is arrow-minimal for s in Γ.

2. If the type S is minimal for s in Γ and $S \nearrow^!_\Gamma \longrightarrow^!_{\beta\top} All(A{\le}S_1{:}K_1)S_2$, then $All(A{\le}S_1{:}K_1)S_2$ is All-minimal for s in Γ.

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

Definition 6.5 (Typechecking algorithm)

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

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

$$\frac{\begin{array}{c} S \nearrow^!_\Gamma \longrightarrow^!_{\beta\top} T_1 \to T_2 \\ \Gamma \vdash_{\mathcal{A}} s \in S \quad \Gamma \vdash_{\mathcal{A}} t \in T \quad \Gamma \vdash_{\mathcal{A}} T \le T_1 \in \star \end{array}}{\Gamma \vdash_{\mathcal{A}} s\,t \in T_2} \qquad \text{(A-Arrow-E)}$$

$$\frac{\Gamma,\, A{\le}T_1{:}K_1 \vdash_{\mathcal{A}} e \in T_2}{\Gamma \vdash_{\mathcal{A}} \mathit{fun}(A{\le}T_1{:}K_1)e \in \mathit{All}(A{\le}T_1{:}K_1)T_2} \qquad \text{(A-ALL-I)}$$

$$\frac{\begin{array}{c} T \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} \mathit{All}(A{\le}T_1{:}K_1)\ T_2 \\ \Gamma \vdash_{\mathcal{A}} t \in T \qquad \Gamma \vdash S \in K \qquad \Gamma \vdash_{\mathcal{A}} S \le T_1 \in K_1 \end{array}}{\Gamma \vdash_{\mathcal{A}} t\, S \in [S/A]T_2} \qquad \text{(A-ALL-E)}$$

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

Theorem 6.6 (Soundness and completeness)

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

This brings us to our final result:

Corollary 6.7 (Decidability of typing) The original typing relation $\Gamma \vdash t \in T$ is decidable.

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

Part III

Conclusions

Chapter 7

Conclusions

7.1 Summary

My thesis proposes and investigates a polarized version of $F^{\omega}_{\leq}$, the higher-order polymorphic λ-calculus with subtyping, similar to the one found in Cardelli [Car90]. It explores its meta-theory, establishing decidability of the system. Including polarized application rules leads to an interdependence of the subtyping and the kinding system. This contrasts with pure $F^{\omega}_{\leq}$, where subtyping depends on kinding but not vice versa. To retain decidability of the system, the equal-bounds subtyping rule is rephrased in the polarized setting as a mutual-subtype requirement. The work here also extends the calculus of Hofmann and Pierce in [HP95b] who focus on polarity information of type variables of the base kind $\star$, not considering type operators in generality, and not investigating type-checking algorithms. To my knowledge this is the first algorithmic account of polarized extensions of pure $F^{\omega}_{\leq}$.

The cycle between subtyping and kinding resembles the situation when so-called bounded operator abstraction (cf. again [Car90]) is included in $F^{\omega}_{\leq}$ or for systems with dependent types. Closer scrutiny reveals differences in the structure of the proofs for decidability for such calculi. For the rest of the chapter I compare the work with recent results in this area, discussing work on bounded operator abstraction, on subtyping dependent types, and on type inference,

7.2 Related and future work

Variance or polarity annotations

In the context of functions over an ordered domain (such as type operators as functions on the domain of types, ordered by subtyping), monotonicity is a natural concept. For different subtyping calculi, the issue of monotonicity has shown up in different guises, especially in connection with binary methods.

Declarative variance annotations on methods In [AC96b] [ACV96] versions of Abadi and Cardelli's object-calculi with *variance* annotations are investigated. The objects in this calculus can be seen as records, supporting method- or record-selection, method-update, and cloning.[1] Another difference to functional records is that late-binding is directly modelled in the semantics. As explained in the section about record calculi in the introduction, the definition of subtyping for object types supporting method update has to be more restrictive than for record types: the possibility of method update requires for object types to insist on identical types for the common fields, disallowing depth-subtyping.

As a consequence, object types of this form do not support method specialization, i.e., when changing to a more refined object carrying a subtype of the original object's type, the types of the methods should reflect this change. This is not feasible if the types of the "inherited" methods are forced to stay identical. To cater for greater flexibility in subtyping, the methods (not the methods' types) of an object type are decorated by variance annotations, where $\circ$ means invariance, $+$ stands for covariance, and $-$ for contravariance. The intuition here differs from ours. The annotations serve to control the allowed run-time manipulations *upon the record* of methods — namely read- or write-access to the record of methods, which is method-selection and -update — and not to describe the way the methods act upon the data of the object. In other words, the annotations are of declarative nature, and not derived structurally, and it is up to the typing system to assure, that the programs conform to the discipline imposed upon them by the variance annotations. For instance, the typing rules have to forbid the update of a method declared as covariant, which means that for methods with read-only access one regains the full depth- plus width-subtyping behavior of ordinary records. The variance notation is lifted to object types with self types, going beyond the simple updatable-records type for objects. This object-calculus can be interpreted [ACV96] in $F_{\leq\mu}$, the polymorphic λ-calculus with subtyping and recursive types.[2] The idea of declarative variance annotations for an object-calculus are also used in [Hic97], where it is extended to a dependent record types and implemented in Nuprl-Light [Hic96].

Variance of row functions In an object-based setting, Fisher [Fis96] (cf. also [FM95]) investigates an object calculus with recursive record types, which, besides method invocation and method override as in the calculi of Abadi and Cardelli, supports method *extension*, the run-time addition of methods to an object. As sketched

[1]Cloning is the creation of a new object from an existing one. This is different from class-based languages, where new objects are instantiated from classes.

[2]For a comparison of this object-encoding with other models known in the literature, see [BCP96].

in the section about record calculi in the introduction, in this situation neither depth- nor width-subtyping is sound, leaving us with no subtyping at all for object types, which is clearly undesirable.

The solution proposed there is to distinguish two phases for objects, one in which both method extension and method override are allowed, and a second one where the structure of the objects is "sealed" and no longer amenable to changes; on the other hand, in this second phase depth- and width-subtyping are allowed. The type system reflects this distinction in separating pro-types for extensible objects (called prototypes) from object types for the fixed ones. This resembles the situation in class-based languages with their strict separation of classes and objects.

But the sealing of prototypes to objects or subtyping between object types is sound only in case the recursion variable does not occur contravariantly inside the record type (pro-types $\mathit{pro}\, A.R$ and object types $\mathit{obj}\ A.R$ are recursive record types). For example, the type of an extensible object modelling a point would be written $\mathit{pro}\, A.\langle\!\langle x{:}\mathit{Int}, \mathit{setx}{:}\mathit{Int} \to A\rangle\!\rangle$, where the type variable A is recursively bound in the body $\langle\!\langle x{:}\mathit{Int}, \mathit{setx}{:}\mathit{Int} \to A\rangle\!\rangle$, a so-called row expression, and *pro* indicates that the methods of values of this type may the extended and overridden. To capture the way the type variable occurs in such a row, *variance annotations* are introduced and integrated into the kinding system, allowing statements of the form $\Gamma \vdash T \in V$ or $\Gamma \vdash R \in V$. There, V is a variance set, which stores variance information for all type variables in R. Similar the variance statements for rows $\Gamma \vdash R \in (V, M)$, where in addition to the variance set, also information about the absence of methods is included in the set M to allow for sound addition of not-yet-present methods. The type system supports also a simple form of type operators, there called row functions, whose variance can be determined by the occurrence of the formal parameter. Row functions in the type system in [Fis96] are restricted in that no higher-order type operators are included, and only types of kind $\star$ and $\star \to^{?} \star$ are supported, to use our notation. Also an ordering on variances and variance sets is introduced, corresponding roughly to our subkinding relation.

The notation and terminology in [Fis96] differs from ours: an operator, there called row function, with invariant variance there means, that it only allows its arguments of be invariant, i.e. identical. Invariant occurrence is denoted by o and corresponds to our $\pm$. The case where a row function ignores its argument is denoted by ?, corresponding to our $\circ$. For monotone and antimonotone appearance of variables there is no ambiguity. Otherwise the rules for calculating variances are, naturally, similar to ours. Since she does not deal with higher-order application in its generality, though, there is no distinction between non-free occurrence of variables and constant occurrence: the distinction becomes visible only in the presence of applying type variables, declared as operators of constant polarity, to types. For example, the type variable A' appears constantly in the type $A\ A'$, if A is declared to be of kind $K_1 \to^{\circ} K_2$.[3] Since in [Fis96], the only functions on type level are row functions of kind $\star \to^{?} \star$, it suffices to consider type variables of base kind $\star$. By the same token, β-reduction on type level is almost trivial.

Positive type operators and binary methods In [HP95b], Hofmann and Pierce present a uniform, type-theoretic account of object-oriented concepts of class-based languages, including encapsulation, subtyping, and message passing. Instead of concentrating on one specific encoding of object-oriented language features, they are interested in a more abstract framework, comprising different known encodings.

In specifically $F^{\omega}_{\leq}$'s object-model [PT94], as one prominent representative fitting in their uniform framework, objects carry existential types to protect the objects' internal states by assuring that they can be accessed via their methods only.[4] Falling back upon the point example, a point object with some internal state for its coordinates and supporting two methods, *getx* and *setx*, to extract the x-coordinate of integer type from the point and to set it, carries the existential type:[5]

$$\mathit{Point} = \exists R{:}\star.R \times \{\mathit{getx}{:}R \to \mathit{Int}, \mathit{setx}{:}R \to \mathit{Int} \to R\}$$

Thus, point objects consist of a state paired with the record of its methods, in this case the two methods *setx* and *getx*, where the existential type only reveals, that there exists some type R for the internal state, but not its nature. The signature of points, to stick with the example, is written as type operator, taking the internal representation type as argument and giving back the record of methods: $\mathit{Fun}(R{:}\star)\{\mathit{getx}{:}R \to \mathit{Int}, \mathit{setx}{:}R \to \mathit{Int} \to R\}$. In $F^{\omega}_{\leq}$'s functional representation of objects — a state paired with the record of methods, and both hidden by an existential type — the methods uniformly take the internal state of type R as first parameter, which means the first parameter can be "factored out" from the interface. This changes the type in the above example to $\mathit{Point} = \exists R{:}\star.R \times R \to \{\mathit{getx}{:}\mathit{Int}, \mathit{setx}{:}\mathit{Int} \to R\}$, with the signature of

$$\mathit{PointSig} = \mathit{Fun}(R{:}\star)\{\mathit{getx}{:}\mathit{Int}, \mathit{setx}{:}\mathit{Int} \to R\},$$

[3]To avoid confusion, I always use the conventions of this thesis, not the ones from [Fis96] or from the works of Abadi and Cardelli.

[4]Existential types to model encapsulation were introduced by Mitchell and Plotkin in [MP88].

[5]The system presented later will not include existentials, but they can be represented in the system. The second order universal type $\forall A.T$ of System F quantifies over all types, *including itself.* This circle, known as impredicativity, makes up the strength of the type system and can be used to encode datatypes [Wra89] not present in the core system; for instance, existential types can be represented as $\exists A.T = \forall A.(\forall B.(T \to B)) \to A$. See Ghelli and Pierce [GP97] for a discussion of especially the encoding of existential types in the presence of the different versions of the bounded All-quantifier subtyping rule in $F_{\leq}$.

instead. So the type of point objects is defined as $Point = \exists R.R{\times}(R \rightarrow PointSig(R))$, or for object types in general $O = \exists R{:}\star.R \times (R \rightarrow Sig(R))$, containing a state of the internal representation type R, which is hidden by the existential quantifier, paired with the methods. To program with objects, it must be possible not only to build objects and give them a type, but also to invoke their methods. This means to transform the internal method implementation of type $R \rightarrow Sig(R)$ for some type R into an externally usable version, so-called *generic* methods, of type $Object(Sig) \rightarrow Sig(Object(Sig))$, where *Object* is the type constructor that builds the existential object type, given a signature *Sig*.

In [HP95b] it is shown that $F^{\omega}_{\leq}$'s object model sketched here allows a uniform definition of generic methods in case the signature is a *positive* type operator, i.e. of kind $\star \rightarrow^{+} \star$ (in [HP95b], positive type operators are denoted by $pos(T)$). This is the case for the signature of points with the *getx* and *setx* method, as the components of the record type behave covariantly and thus $PointSig \in \star \rightarrow^{+} \star$. In contrast, the signature of point objects with an equality method, comparing the states of two points,

$$PointSig' = Fun(R{:}\star)\{getx{:}Int, setx{:}Int \rightarrow R, eq{:}R \rightarrow bool\}$$

is not positive — the type variable R occurs negatively, on the left-hand side of the arrow, in type of the *eq*-method — but carries the weaker kind $\star \rightarrow^{\pm} \star$. Here, *eq* is one standard example for a so-called *binary* method, binary, as it requires access to two the internal representation of different objects of the same class.

To be able to characterize monotone signatures that allow the uniform definition generic methods as above, Hofmann and Pierce extend $F^{\omega}_{\leq}$ (or rather a fragment of it) with a predicate *pos* on type operators of kind $\star \rightarrow \star$. In fact, they do not only characterize the circumstances under which generic methods exist, but give the uniform construction by "lifting" a function $f \in S_1 \rightarrow S_2$ through a type T, yielding a function $lift_T^{f \rightarrow A}$ of type $[S_1/A]T \rightarrow [S_2/A]T$, in case the variable A occurs only monotonely inside T. In the presence of monotonicity (and anti-monotonicity) information about type operators, besides pointwise subtyping, also monotone and antimonotone subtyping is allowed.

$$\frac{\Gamma \vdash T \in \star \rightarrow \star \qquad pos(T) \qquad \Gamma \vdash U_1 \leq U_2}{\Gamma \vdash T\,U_1 \leq T\,U_2} \quad \text{(S-Pos-Mono)}$$

Our definition of variable occurrence in Section 5.3 is a generalization of the definition in [HP95b]. First of all, like in Fisher's system, the definition of *pos* only handles type variables of the base kind $\star$, and polarized operators of kind $\star \rightarrow \star$, which corresponds to a fragment of F^3 with subtyping.[6] This restriction makes it unnecessary to consider the positive or negative occurrence of type variables in applications. Another difference is that Hofmann and Pierce do not consider the additional polarities $\pm$ and $\circ$. As already mentioned in the comparison with Fisher's work, in restricting the type variables to the base kind $\star$, there is no need to introduce constant variable occurrence, because it coincides with the fact that the variable does not occur free at all, and likewise there is no need for the non-variant application rule S-App$\pm$. It's only in the presence of polarized application in the higher order case that the concepts of constant occurrence and "no free occurrence" of a type variable split; likewise the notions of $\beta\top$-equality and equivalence of types. Consequently the positive occurrence of a type variable A in a universally quantified type $All(A'{\leq}T_1)T_2$ depends on the type variable A not occurring freely inside the upper bound T_1, lest to destroy the Kernel Fun subtyping rule for All-types. In our higher-oder setting, we will have to rephrase the equal-bounds subtyping rule in the polarized setting requiring mutual subtype relationship between the upper bounds of the All-types, and instead of insisting on the type variable not occurring at all, we require only constant appearance, which is weaker.

Type inference Monotonicity information for type variable or for type operators seems also relevant for type *inference*. By reconstructing (or trying to reconstruct) omitted types from the context, type inference can free the programmer from explicitly feeding a type as parameter to a polymorphic function, or from annotating the type of the formal parameter of a function. Recently, in a couple of papers Pierce and Turner [PT98] [PT97a] [Pie97a] investigate partial, local type-inference for a Kernel-Fun variant of $F_{\leq}$ including a bottom type as the dual to the more common maximal type *Top*. Considering the case of inferring an omitted type argument of a polymorphic function, the types to be inferred are represented by type variables and at a certain stage during the inference process, the algorithm has to calculate a substitution of these variables, such that the result after the substitution is minimal and where the type replacing the variable is drawn from an interval in the subtyping hierarchy (These "interval"-constraints for the omitted type arguments are collected at an earlier stage of the algorithm). Obviously, the choice of the replacement is determined by the way the variable it replaces occurs inside the type; for example, if the type variable occurs monotonely, choosing the minimal substitution satisfying the constraint yields a minimal result. Their type inference algorithm can also handle type operator constants in case they are *monotone*, for instance, the *List* type constructor of kind $\star \to^{+} \star$. Lacking is support for user-definable type-operators as in the higher-order $F^{\omega}_{\leq}$. The system of polarized higher-order subtyping presented

[6]For an introduction to the hierarchy of polymorphic calculi, from the polymorphic λ-calculus F or F^2 to the full higher-order case of F^{ω} see [PDM89].

here could be helpful in the generalization. Currently, Pierce and Turner experiment with various extension for type inference for Pict, a π-calculus based programming language whose type system is based on $F^{\omega}_{\leq}$.

Subtyping dependent types $F^{\omega}_{\leq}$ is an impredicative type theory, where terms depend on terms and types, and types take types as parameters (in which case they are called type operators). Systems where types can depend on terms are said to have *dependent types*. This allows to express functions, uniformly parametric for a range of terms.

Recently, interest in calculi with dependent types extended by subtyping has started. Aspinall and Compagnoni [AC96c] investigate $\lambda P_{\leq}$, a first order calculus with dependent types (known as λP or $\lambda\Pi$) extended by subtyping, establishing decidability of the system. Chen [Che97a] studies a variant of this system, which he calls $\lambda\Pi_{\leq}$.[7]

With dependent types, the ordinary, non-dependent function types $S \to T$ generalize to dependent function types of the form $\Pi x{:}S.T$, characterizing function with domain S, whose range T may depend on the term parameter x of type S. The non-dependent function space is a proper type and carries the kind $\star$. Allowing dependent function types, the kinding systems has to be extended to assure that only terms of the appropriate type are fed into the dependent function type constructor:

$$\frac{\Gamma, x{:}S \vdash T \in K}{\Gamma \vdash \Lambda x{:}S.T \in \Pi x{:}T.K} \qquad \text{(K-Pi)}$$

Correspondingly there is an elimination rule for the Π-kinds, controlling the application of a dependent function type to a term:

$$\frac{\Gamma \vdash T \in \Pi x{:}S.K \qquad \Gamma \vdash t \in S}{\Gamma \vdash T\ t \in [t/x]K} \qquad \text{(K-App)}$$

The conclusion of this kinding rule depends on the derivation of a typing statement. In the presence of subtyping, i.e., subsumption, this leads to the following mutual dependence:

This is in contrast to polarized $F^{\omega}_{\leq}$, where the polarized application rules entails an interdependence of Figure 7.2.

Thus, the dependent type systems $\lambda P_{\leq}$ and $\lambda\Pi_{\leq}$ are more complicated, since the mutual dependence concerns three families of statements, not just two. One the

[7]The systems $\lambda\Pi_{\leq}$ and $\lambda P_{\leq}$ are different formulations of the same theory. They can be regarded as an extension of Pfenning's refinement type system [Pfe93b] for the Logical Framework. Pfenning's subsort relation is more restricted than the full subtyping relation in that only trivial subtyping is allowed between type families and reflexivity of the subsorting relation insists on identity, not on β-convertibility.

$$\begin{tikzcd}
\text{kinding} \arrow[dr, "\text{dep. types}"'] & & \text{subtyping} \arrow[ll, bend right=10] \\
& \text{typing} \arrow[ur, "\text{subsumption}"'] &
\end{tikzcd}$$

Figure 7.1: First-order dependent types

$$\begin{tikzcd}
\text{kinding} \arrow[rr, bend left=5, "\text{pol. application}"] & & \text{subtyping} \arrow[ll, bend left=5] \\
& \text{typing} \arrow[ur, "\text{subsumption}"'] &
\end{tikzcd}$$

Figure 7.2: Higher-order polarized subtyping

other hand, they are simpler in one important respect: as first-order systems, unlike $F^{\omega}_{\leq}$ or $F^{\omega}_{\wedge}$, they do not have universally quantified types, let alone All-types with subtype-bounded variables. This means, no rule in the system *extends* the contexts by new non-trivial subtype declarations of the form $A{\leq}T{:}K$, making, for example, the proof of strong normalization a lot easier.

At the highest level, both Aspinall/Compagnoni and Chen address the problem of the interdependence of the statements the same way: As it is not possible to dispense with the subsumption rule or with the dependence of kinding upon typing — both forming the essence of the system — they break the link between subtyping and kinding. Of course at the end they have to make sure that this does not allow to derive subtype relation between ill-kinded types.

On a more detailed scale, there are a some differences between both works. Aspinall and Compagnoni prove admissibility of cut in their *algorithmic* system, which operates exclusively on normalized types, whereas Chen performs cut-elimination (or admissibility of cut) not in a normalizing system, but allows types of arbitrary form.

Another complication not present in the systems for $F^{\omega}_{\leq}$ concerns subject reduction. Since kinding depends on typing, subject reduction for kinding depends on subject reduction for typing. Both works address the problem by separating β-reduction into reduction on term level and reduction on type-level (substituting term into a type), treating reduction on types first, while proving general subject reduction after cut-elimination.

Comparing the two figures 7.1 and 7.2 for $\lambda\Pi_{\leq}$ and for polarized $F_{\leq}^{\omega}$, the obvious question is whether we could use the approach of Aspinall and Compagnoni, resp. Chen, to break the cycle, i.e., instead of introducing an additional set of statements for variable occurrence $\Gamma \vdash T\ ?_A$ in order to separate typing and subtyping, would it be possible to render subtyping not depending on kinding instead?[8] The answer is no. The essence of polarized subtyping are the application rules, where the choice of rule crucially depends on the *specific* polarized kind of the type operator, which cannot be determined beforehand. This is different from the situation for $\lambda P_{\leq}$, $\lambda\Pi_{\leq}$ (or pure $F_{\leq}^{\omega}$) where for a start one can ignore kinding in subtyping derivations, knowing that the subtyping derivations does not destroy well-kindedness and thus, beginning with well-kinded types, the types of each subtyping statement justified by a subderivation carries *some* kind (which is, without subkinding, moreover unique and coincides for each pair of types).

An extension of the work on $\lambda\Pi_{\leq}$ is carried out in [Che97b], establishing decidability of a subtyping extension of the full calculus of constructions, called $\lambda C_{\leq}$. This system, though, does not contain bounded quantification nor subkinding. Compared to $\lambda P_{\leq}$, the situation gets even more involved, giving altogether four different β-reduction relations. What still is simpler than in $F_{\leq}^{\omega}$, $F_{\wedge}$, or even $F_{\leq}$ is that the subtyping extension of the calculus of constructions does not contain the notion of bounded quantification. The issue of subtyping of dependent types is further discussed in [CL96].

Another extension (called $\lambda_{\leq\{\}}$) of the first-order typed λ-calculus by an especially simple form of dependent types, so called singleton types, is presented by Aspinall in [Asp95]; similarly for the second-order case, incorporating singleton types into $F_{\leq}$, in [Hay94]. Luo [Luo96] studies subtyping based on coercions in the framework of his UTT [Luo92], the unified theory of types, i.e., the calculus of constructions with type universes and inductive types, implemented in the Lego proof assistant [LPT89] [leg97]. Proof-reuse via subtyping is investigated in [Luo95]. An implementation of subtyping in Lego is currently under way [Bai96].

Bounded operator abstraction The formulation of $F_{\leq}^{\omega}$ investigated in this thesis, whether polarized or not, contains an asymmetry concerning the bindings of type variables. Whilst the type variable in the bounded quantifier type ranges over all subtypes of its upper bound, no such restriction is expressible for the type operators (More exactly, in operators $\mathit{Fun}(A{:}K)T$ the argument ranges over all subtypes of the trivial upper bound $\mathit{Top}(K)$).

It is relative straightforward to formulate a system relaxing this constraint, allow-

[8]The separation of different forms of β-reduction is not an issue here, since for $F_{\leq}^{\omega}$, reduction on term level can be performed separately.

ing operators with non-trivial upper bound $\mathit{Fun}(A{\leq}T_1{:}K_1)T_2$. Indeed, it was in this more general form, that $F^\omega_\leq$ appeared in Cardelli's note [Car90]. As a consequence, also the kinding system has to be adapted. The kind of a type operator with bounded abstract type parameter now has to capture the fact that no longer all types of kind K_1 are legitimate arguments for the operator, as expressed by the kind $K_1 \to K_2$, but only those less or equal type T_1:

$$\frac{\Gamma, A{\leq}T_1{:}K_1 \vdash T_2 \in K_2}{\Gamma \vdash \mathit{Fun}(A{\leq}T_1{:}K_1)T_2 \in \Pi A{\leq}T_1{:}K_1.K_2} \qquad \text{(K-Pi-I)}$$

The corresponding elimination rule (replacing K-Arrow-E) then reads:

$$\frac{\Gamma \vdash S \in \Pi A{\leq}T_1{:}K_1.K_2 \qquad \Gamma \vdash T \leq T_1 \in K_1}{\Gamma \vdash S\;T \in [T/A]K_2} \qquad \text{(K-Pi-E)}$$

Further the subtype system gets more complicated, which can be seen at the rule for pointwise application:

$$\frac{\Gamma \vdash S \leq T \in \Pi A{\leq}V{:}K_1.K_2 \qquad \Gamma \vdash U \leq V \in K_1}{\Gamma \vdash S\;U \leq T\;U \in [V/A]K_2} \qquad \text{(S-App)}$$

Compagnoni and Goguen [CG97] study a system like this, proving decidability of $F^\omega_\leq$ with bounded operator abstraction (without subkinding, without polarities, and of course using the equal-bounds subtyping rule for All-types). The rule K-Pi-E for the application of subtype bounded operators renders kinding dependent upon subtyping, similar to the situation for dependent types, only that for $\lambda\Pi_\leq$ and $\lambda P_\leq$, kinding depends on subtyping via the typing statements, whereas bounded operator abstraction directly leads to the cycle. The situation is sketched in Figure 7.3.

$$\begin{tikzcd}
\text{kinding} \arrow[rr, bend left=5, "\text{bounded abstr.}"] & & \text{subtyping} \arrow[ll, bend left=5] \\
& \text{typing} \arrow[ur, "\text{subsumption}"'] &
\end{tikzcd}$$

Figure 7.3: Bounded operator abstraction

So the picture very much looks like the one for polarized $F^\omega_\leq$ in Figure 7.2, but the source of the interdependence now is bounded operator abstraction, not polarized application.

The approach pursued in this thesis, namely breaking the cycle by rendering kinding independent of subtyping is out of question here, as the very nature of bounded operator abstraction requires this dependence in rule K-Pi-E.

Instead, in an elegant proof Compagnoni and Goguen handle kinding and subtyping at the same time. They introduce a *typed operational semantics* to get the metatheoretical results of cut-elimination, subject reduction, and decidability. The proof is an adaption of the one in [Gog95], where the typed operational semantics is used for strong termination of the simply typed λ-calculus with dependent types, lifted on the type level and including subtyping. The core of the argument, which they call soundness, is a strong normalization proof à la Tait and Martin-Löf [Tai67][ML75].[9] The typed operational semantics can be understood as a derivation system combining subtyping statements, kinding statements, and normal-form reduction.[10] The typed operational semantics is a rough analogue to the algorithmic formulations of subtyping in [Com95b][PS97] which corresponds to the strong cut-free system here, and where due to the dependence of kinding upon subtyping, the reduction relation has to be incorporated to the premises of the kinding system, as well. (For a more detailed account of typed operational semantics, see Goguen [Gog94b] [Gog94a] [Gog95].) Unlike [CG97], our system allowed to address kinding (and equivalence) and subtyping one after the other by stratifying kinding and subtyping.

Like our system, $F^\omega_\leq$ with bounded operator abstraction does not enjoy a unique kinding property, but for a different reason. As kinds contain types, β-conversion on kind level has to be considered; but types do have a unique normal form kind. On the other hand, the lack of unique-kinding for polarized $F^\omega_\leq$ stems from subkinding in addition to type conversion, and kinding uniqueness has to be replaced by a minimal kinding property.

We can draw the parallel further to the subtyping systems. As we have seen in the development in Part II of the thesis, considering polarity information means that $\beta\top$-equality is no longer an adequate notion of equality on type level. Instead we used mutual subtyping relation $\Gamma \vdash S \gtreqless T \in K$, which we were able to replace in the algorithm by the stricter notion of *equivalence* on types, denoted by $\Gamma \vdash S \equiv T \in K$. So the correspondence is the following: in the polarized setting, $\Gamma \vdash S \gtreqless T \in K$

[9]The soundness proof of Compagnoni and Goguen corresponds to the completeness-part of our proof. Their choice of the term is justified, in that they take the typed-operational semantics — corresponding roughly to our $\vdash_{\mathcal{CS}}$-system — as starting point and prove the soundness of the original formulation with respect to this semantics. Here, we take the dual perspective, proving soundness and completeness of the algorithm against the original formal system.

[10]In effect, it is somewhat refined in that it not only considers normal forms, but also a variant of weak-head normal forms at the same time. The variant of weak-head normal forms insists that for types of the form $A\ T_1\ \ldots T_n$ the T_i's are already in normal form, to deal with the problematic cases of application and promotion.

plays the role of β-equivalence for the calculus of bounded operator abstraction, while equivalence replaces identity.

Polarized bounded operator abstraction The obvious question is: is it possible to combine monotonicity information with bounded operator abstraction?

So, consider the kinding and subkinding relations. For bounded operator abstraction, kinds contains types, which means that well-formedness of kinds can no longer be determined by the structure of the kind alone, and statements of the form $\Gamma \vdash K\ \mathit{kind}$ have to be included. By the same argument, the subkinding relation has to be derived with respect to a context, yielding a system for $\Gamma \vdash K_1 \leq K_2$. While we are at it, we can go a step further beyond the form of subkinding which merely lifts the ordering on polarities to the level of kinds, and take also the subtype-bound of the operator into consideration, stipulating that an operator with a less restrictive bound lies above one that can handle a larger range of types as arguments:

$$\frac{\begin{array}{c}\Gamma \vdash K_1' \leq K_1 \qquad ? \leq ?' \\ \Gamma \vdash T' \leq T \in K_1' \qquad \Gamma, A{\leq}T'{:}K_1' \vdash K_2 \leq K_2'\end{array}}{\Gamma \vdash \Pi A^{?}{\leq}T{:}K_1.K_2 \leq \Pi A^{?'}{\leq}T'{:}K_1'.K_2'} \quad \text{(K-Sub-Pi)}$$

In a similar form, subkinding — without polarities — was proposed in Pierce and Pollack's formulation [PP92]. (See also [Car90], without an explicit rule of subsumption for subkinding) In the presence of polarized subkinding, the kinding system has to be adapted, too, incorporating for example a rule for monotone subtype-bounded type operators:

$$\frac{\begin{array}{c}\Gamma, A{\leq}T_1{:}K_1 \vdash T \in K_2 \\ \Gamma, A_2{\leq}T_1{:}K_1, A_1{\leq}A_2{:}K_1 \vdash [A_1/A]T_2 \leq [A_2/A]T_2 \in K_2\end{array}}{\Gamma \vdash \mathit{Fun}(A{\leq}T_1{:}K_1)T_2 \in \Pi A^{+}{\leq}T_1{:}K_1.K_2} \quad \text{(K-Pi-I+)}$$

This leads to a considerable complication of the polarized system as well as of the one investigated by Compagnoni and Goguen; the graph of dependence sketched in the following figure (the arrow from subkinding to subtyping is caused be rule K-Sub-Pi):

As mentioned, it is well-known that the full $F^{\omega}_{\leq}$ subtyping rule for All-types has quite a number of unpleasant consequences for the meta-theory of the system, including undecidability of even $F_{\leq}$ in the second-order case. This had led us to consider a polarized variant of the equal-bounds quantifier rule, requiring the bounds of the universally quantified types under comparison to be in mutual subtype relation (cf. S-All).

Lest to face similar problems with subkinding of bounded operator abstraction, it seems natural to impose an analogous discipline for bounded operator abstraction, as well, using a rule S-Abs-Kernel

$$\begin{tikzcd}
 & \text{subkinding} \arrow[dr] & \\
\text{kinding} \arrow[ur, bend left, "\text{subsumption}"] \arrow[rr, bend left=10, "\text{bounded abstr.}"] & & \text{subtyping} \arrow[ll, bend left=10] \\
 & \text{typing} \arrow[ur, bend right, "\text{subsumption}"'] &
\end{tikzcd}$$

Figure 7.4: Polarized bounded operator abstraction

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{Fun}(A{\leq}S_1{:}K_1)S_2, \mathit{Fun}(A{\leq}T_1{:}K_1)T_2 \in \Pi A^?{\leq}S_1{:}K_1.K_2 \\ \Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \leq T_2 \in K_2 \quad \Gamma \vdash S_1 \gtrless T_1 \in K_1\end{array}}{\Gamma \vdash \mathit{Fun}(A{\leq}S_1{:}K_1)S_1 \leq \mathit{Fun}(A{\leq}T_1{:}K_1)T_2 \in \Pi A^?{\leq}S_1{:}K_1.K_2}$$

instead of the more general S-ABS:[11]

$$\frac{\begin{array}{c}\Gamma \vdash \mathit{Fun}(A{\leq}S_1{:}K_1)S_2, \mathit{Fun}(A{\leq}T_1{:}K_1)T_2 \in \Pi A^?{\leq}S_1{:}K_1.K_2 \\ \Gamma, A{\leq}T_1{:}K_1 \vdash S_2 \leq T_2 \in K_2 \qquad \Gamma \vdash T_1 \leq S_1 \in K_1\end{array}}{\Gamma \vdash \mathit{Fun}(A{\leq}S_1{:}K_1)S_1 \leq \mathit{Fun}(A{\leq}T_1{:}K_1)T_2 \in \Pi A^?{\leq}S_1{:}K_1.K_2} \quad \text{(S-ABS)}$$

The meta-theoretical investigations of subtyping calculi, partly discussed here, all are sensitive in a delicate way to the order in which the proofs of subject reduction for kinds and for subtyping, of cut-elimination or admissibility of cut, and of strong termination are performed. The most elegant and insightful treatment of these issues so far is the one by Compagnoni and Goguen, using typed operational semantics. Hence it is natural to proceed in this direction.

The foremost complication I expect is the need to perform polarized substitution. The proof in [CG97] proceeds in the following order: first admissibility of cut in the typed operational semantics, which corresponds to the strong, cut-free derivations in this thesis, and *afterwards* (as part of the "soundness"-proof), preservation of subtyping under substitution. Actually, this order seems impossible in the presence of polarized subtyping. To pass in review the stages in which we showed cut-elimination and subject reduction in the stratified system, the proof of admissibility of cut (Proposition 5.83) already depends on subject reduction and preservation of subtyping under (polarized) substitution, whereas in [CG97], the proof can be carried

[11]The name K-ABS-KERNEL for this rule is chosen in reminiscence of the equal-bounds subtyping rule for universally quantified types of Cardelli and Wegner's Kernel Fun.

out independently.[12] In both cases, admissibility of cut is shown in the system $\vdash_{\mathcal{CS}}$.[13] The obstacle once more are the polarized application rules. Recall that the critical case for cut-elimination was R-Promote used to promote the cut-type:

$$\frac{U \longrightarrow^!_{\beta\top} A\ U_1 \ldots U_n \uparrow_\Gamma \Gamma(A)\ U_1 \ldots U_n \quad \Gamma \vdash_{\mathcal{CS}} \Gamma(A)\ U_1 \ldots U_n \le T \in K}{\Gamma \vdash_{\mathcal{CS}} U \le T \in K}$$

We were able to solve this case by the analysis of strong, cut-free derivations in Section 5.11. One important step in this proof was Lemma 5.80 on page 91, stating that under appropriate assumptions about kinding and polarity, $\Gamma \vdash_{\mathcal{CS}} A\ S_1 \ldots S_n \le A\ T_1 \ldots T_n \in K$ implies $\Gamma \vdash_{\mathcal{CS}} \Gamma(A)\ S_1 \ldots S_n \le \Gamma(A)\ T_1 \ldots T_n \in K$. The proof of this lemma used a property like the following:

Assume $\Gamma \vdash_{\mathcal{CS}} S \le T \in K_1 \to^+ K_2$. If $\Gamma \vdash_{\mathcal{CS}} U_1 \le U_2 \in K_1$, then $\Gamma \vdash_{\mathcal{CS}} S\ U_1 \le T\ U_2 \in K_2$. (1)

In the light of Compagnoni's and Goguen's insight, this corresponds to one crucial step in Tait's method. Remember that this proof technique proceeds by induction on the structure of types, thus here by induction on the types of types, called kinds, and the "application" of $\Gamma \vdash_{\mathcal{CS}} S \le T \in K_1 \to^+ K_2$ to $\Gamma \vdash_{\mathcal{CS}} U_1 \le U_2 \in K_2$ yields a statement, whose kind K_2 is (structurally) smaller than $K_1 \to^+ K_2$.[14] The use of Tait's method (the soundness proof in [CG97]) *depends* on the proof of cut-elimination, something we are just about to show in the polarized setting.

Note also that without polarized subtyping the promotion case is not really an issue. As explained in Section 3.5, the case of promotion in the proof of cut-elimination can be solved without induction and also without the above implication, namely by the simple observation that $\Gamma \vdash_{\mathcal{CS}} S \le A\ U_1 \ldots U_n \in K$ implies $S \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\ U_1 \ldots U_n$. Hence the polarized version of subtyping introduces an interdependence into the proof, not present in the pure case of bounded operator abstraction and which does not lend itself readily to the order in which Compagnoni and Goguen proceed, following Tait's method.

[12]The subject reduction lemma in [CG97] is not subject reduction in its full meaning in that it does not involve preservation of subtyping under reduction.

[13]For the sake of discussion, I will identify the system $\vdash_{\mathcal{CS}}$ from this thesis with the typed operational semantics from [CG97], even if there are differences; most notably, no operational behavior is included in the *kinding* relation here as in the typed operational semantics. In [CG97], the derivations with the typed operational semantics are written $\vdash_S$, with S for "semantics", not for "strong" derivations.

[14]For material about Tait's method, or reducibility method, consult Tait [Tai67][Tai75] or Girard [Gir71][Gir72]. It has been widely used to establish for example strong-termination results for typed λ-calculi.

Another way to attack polarized bounded operator abstraction is to try to adapt the proof structure of this thesis. As just explained, we could not help but prove preservation of subtyping under substitution first, leading to preservation of subtyping under reduction, and finally to the proof that replacing ordinary reduction of the cut-free reducing system by normalizing reduction in the $\vdash_{\mathcal{CS}}$-system does not loose any expressiveness, where the last step already required termination of $\beta\top\Gamma$-reduction.

Now, the obstacle is the polarized *substitution* property (cf. Lemma 5.58 on page 83), more specifically the case of promotion. Considering the monotone case, we are given $\Gamma \vdash_{\mathcal{CS}} A'\ S_1 \dots S_n \leq T \in K$ by a derivation ending with an instance of R-PROMOTE and we are to show preservation of substitution concerning a type variable A occurring monotonely in $A'\ S_1 \dots S_n$ or T. The interesting case is, where the monotonicity assumption concerns the type on the left, i.e., we are given $\Gamma \vdash A'\ S_1 \dots S_n\ +_A$. What happens if $A = A'$? Well, it depends on whether we consider bounded operator abstraction or not. In the easier case, which we explored in this thesis, A is bounded by the trivial upper bound $\mathit{Top}(K)$. Hence we know by the premise of the promotion rule $\Gamma \vdash_{\mathcal{CS}} \mathit{Top}(K')\ S_1 \dots S_n \leq T \in K$, which directly entails that T itself reduces to a maximal type, from which the case follows. In the general case of bounded operator abstraction, we are given $\Gamma(A) = S_0$ for an arbitrary type S_0 and thus $\Gamma \vdash_{\mathcal{CS}} S_0\ S_1 \dots S_n \leq T$ by subderivation. Since the type variable A cannot occur free in S_0, we easily obtain by induction

$$\Gamma' \vdash_{\mathcal{CS}} S_0\ [U_1/A]S_1 \dots [U_1/A]S_n \leq [U_2/A]T \in K.$$

From $\Gamma \vdash_{\mathcal{CS}} A \leq S_0 \in K_0$ follows

$$\Gamma \vdash_{\mathcal{CS}} A\ [U_1/A]S_1 \dots [U_1/A]S_n \leq S_0\ [U_1/A]S_1 \dots [S_n/A]S_n \in K$$

by the implication (1) from above and further $\Gamma \vdash_{\mathcal{CS}} A\ [U_1/A]S_1 \dots [S_n/A]S_n \leq [U_2/A]T \in K$ by *transitivity.* Once again we are stuck: both transitivity elimination and property (1) depend on the substitution lemma, not as before, vice versa.

To deal with R-TRANS, we could try to prove the substitution lemma not in the cut-free system $\vdash_{\mathcal{C}}$, hence we do not to bother about the admissibility of cut at this point yet. However, as mentioned already in the in outline of proof in Section 5.2, the form of the application rules requires that — to stick with the monotone case — the type variable replaced is required to occur monotonely in one of the two types. Now the rule of transitivity introduces a new meta-variable, the cut-type, for which the occurrence of the variable replaced is unknown and the induction will fail. For pointwise substitution it doesn't matter, in which way the cut type contains the variable under consideration, if at all, so pure $F^{\omega}_{\leq}$ should allow to proceed in this order also in the presence of bounded operator abstraction.

Summing up it appears as if bounded operator abstraction and polarized application extend $F^{\omega}_{\leq}$ in two different and not orthogonal ways. Decidability of polarized $F^{\omega}_{\leq}$ with bounded operator abstraction, giving the full system of Cardelli's note[15] [Car90], remains open.

[15]modulo K-Sub-Kernel.

Bibliography

[Aba94] Martín Abadi. Baby Modula-3 and a theory of objects. *Journal of Functional Programming*, 4(2):249–283, April 1994. An earlier version appeared as DEC Systems Research Center Research Report 95, (February, 1993).

[AC93] Roberto M. Amadio and Luca Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993. A preliminary version appeared in [ACM91] (pp. 104–118) and as DEC Systems Research Center Research Report number 62, August 1990.

[AC94] Martín Abadi and Luca Cardelli. A theory of primitive objects. untyped and first-order systems. In Hagiya and Mitchell [HM94], pages 296–320.

[AC96a] Martín Abadi and Luca Cardelli. On subtyping and matching. *ACM Transactions on Programming Languages and Systems*, 18(4):401–423, July 1996. A previous version appeared in [Olt95].

[AC96b] Martín Abadi and Luca Cardelli. *A Theory of Objects.* Monographs in Computer Science. Springer, 1996.

[AC96c] David Aspinall and Adriana Compagnoni. Subtyping dependent types. In *Eleventh Annual Symposium on Logic in Computer Science (LICS).* IEEE, Computer Society Press, July 1996.

[ACM86] ACM. *Object Oriented Programing: Systems, Languages, and Applications (OOPSLA) '86*, 1986. in *SIGPLAN Notices* 21(11).

[ACM88a] ACM. *Conference on Lisp and Functional Programming*, July 1988.

[ACM88b] ACM. *Fifteenth Annual Symposium on Principles of Programming Languages (POPL)*, January 1988.

[ACM90] ACM. *Seventeenth Annual Symposium on Principles of Programming Languages (POPL)*, January 1990.

[ACM91] ACM. *Eighteenth Annual Symposium on Principles of Programming Languages (POPL)*, January 1991.

[ACM92] ACM. *Nineteenth Annual Symposium on Principles of Programming Languages (POPL)*, January 1992.

[ACV96] Martín Abadi, Luca Cardelli, and Ramesh Viswanathan. An interpretation of objects and object types. In *Proceedings of POPL '96*, pages 396–409. ACM, January 1996.

[AFM97] Ole Agesen, Stephen N. Freund, and John C. Mitchell. Adding type parameterization to the Java language. Technical report, Sun Microsystems Laboratories and Standford University, January 1997.

[AG96] Ken Arnold and James Gosling. *The Java Programming Language*. Addison-Wesley, 1996.

[AM97] M. Akşit and S. Matsuoka, editors. *ECOOP '97*, volume 1241 of *Lecture Notes in Computer Science*, 1997.

[Ama91] Roberto M. Amadio. Recursion over realizability structures. *Information and Computation*, 90(2):55–85, 1991.

[Ame89] Pierre America. Issues in the design of a parallel object-oriented language. *Formal Aspects of Computing*, 1(4):366–411, 1989.

[AP90] Martín Abadi and Gordon D. Plotkin. A PER model of polymorphism and recursive types. In *Fifth Annual Symposium on Logic in Computer Science (LICS) (Philadelphia, PA)* [IEE90], pages 355–365.

[Asp95] David Aspinall. Subtyping with singleton types. In Pacholski and Tiuryn [PT95a].

[Bai96] Antony Bailey. Lego with implicit coercions (draft). Technical report, Department of Computer Science, University of Manchester, April 1996. Available through `http://www.cs.man.ac.uk/~baileya/research.html`.

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

[Bar92] Henk P. Barendregt. Lambda calculi with types. In Samson Abramsky, Dov Gabbay, and Thomas Maibaum, editors, *Handbook of Logic in Computer Science*, volume 1: Mathematical Structures, pages 117–309. Oxford University Press, 1992.

[BCC+96] Kim B. Bruce, Luca Cardelli, Giuseppe Castagna, the Hopkins Objects Group (Jonathan Eifrig, Scott Smith, Valery Trifonov), Gary T. Leavens, and Benjamin Pierce. On binary methods. *Theory and Practice of Object Systems*, 1(3):221–242, 1996.

[BCGŠ91] Val Breazu-Tannen, Thierry Coquand, Carl Gunter, and André Ščedrov. Inheritance as implicit coercion. *Information and Computation*, 93:172–221, 1991. Also in the collection [GM94].

[BCP96] Kim Bruce, Luca Cardelli, and Benjamin Pierce. Comparing object encodings. In Bruce [Bru96a]. Summary.

[BDMN79] Graham M. Birtwistle, Ole-Johan Dahl, Bjorn Myhrhaug, and Kristen Nygaard. *Simula Begin.* Studentlitteratur (Lund, Sweden), Bratt Institute Fuer Neues Lerned (Goch, FRG), Chartwell-Bratt Ltd (Kent, England), 1979.

[BFP97] Kim B. Bruce, Adrian Fiech, and Leaf Petersen. Subtyping is not a good "match" for object-oriented languages. In Akşit and Matsuoka [AM97].

[BHJ+87] Andrew Black, Norman Hutchinson, Eric Jul, Henry Levy, and Larry Carter. Distribution and abstract types in Emerald. *IEEE Transactions on Software Engineering*, SE-13(1):65–76, January 1987.

[BL90] Kim B. Bruce and Giuseppe Longo. A modest model of records, inheritance, and bounded quantification. *Information and Computation*, 87:196–240, 1990. Also in the collection [GM94]. An earlier version appeared in the proceedings of the IEEE Symposium on Logic in Computer Science, 1988.

[BLM96] J. Bank, B. Liskov, and A. Myers. Parameterized types and Java. Technical Report MIT/LCS/TM-553, Massachusetts Institute of Technology, May 1996.

[BM92] Kim Bruce and John Mitchell. PER models of subtyping, recursive types and higher-order polymorphism. In *Nineteenth Annual Symposium on Principles of Programming Languages (POPL) (Albuquerque, NM)* [ACM92], pages 316–327.

[Boe85] H.-J. Boehm. Partial polymorphic type inference is undecidable. In *Twenty-sixth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 339–345. IEEE, Computer Society Press, October 1985.

[Bru92] Kim Bruce. The equivalence of two semantic definitions for inheritance in object-oriented languages. In S. Brookes, M. Main, A. Melton, M. Mislove, and D. Schmidt, editors, *Mathematical Foundations of Programming Semantics 1991*, volume 598 of *Lecture Notes in Computer Science*, pages 102–123. Springer, 1992.

[Bru94] Kim B. Bruce. A paradigmatic object-oriented programming language: Design, static typing and semantics. *Journal of Functional Programming*, 4(2), April 1994. A preliminary version appeared in POPL 1993 under the title "Safe Type Checking in a Statically Typed Object-Oriented Programming Language", and as Williams College Technical Report CS-92-01.

[Bru96a] Kim Bruce, editor. *Informal Proceedings of the Third International Workshop on Foundations of Object-Oriented Languages (FOOL'96)*, August 1996. Available electronically through `http://www.cs.williams.edu/∼kim/FOOL/Abstracts.html`.

[Bru96b] Kim Bruce. Typing in object-oriented languages: Achieving expressibility and safety. Technical report, Williams College, September 1996. Available through http://www.cs.williams.edu/∼kim, to appear in *Computing Surveys*, 1997.

[BSvG93] Kim Bruce, Angela Schuett, and Robert van Gent. TOIL, a new type-safe, object-oriented imperative language. Technical report, Williams College, 1993.

[BSvG95] Kim Bruce, Angela Schuett, and Robert van Gent. PolyTOIL: A type-safe polymorphic object-oriented language. In Olthoff [Olt95], pages 27–51.

[C$^+$86] R. L. Constable et al. *Implementing Mathematics with the NuPRL Proof Development System.* Prentice Hall, 1986.

[Car87] Luca Cardelli. Basic polymorphic typechecking. *Science of Computer Programming*, 8:147–172, 1987.

[Car88a] Luca Cardelli. A semantics of multiple inheritance. *Information and Computation*, 76:138–164, 1988. A preliminary version appeared in [KMP84].

[Car88b] Luca Cardelli. Structural subtyping and the notion of power types. In *Fifteenth Annual Symposium on Principles of Programming Languages (POPL) (San Diego, CA)* [ACM88b], pages 70–79.

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

[Car91] Luca Cardelli. Typeful programming. In E. J. Neuhold and M. Paul, editors, *Formal Description of Programming Concepts.* Springer, 1991. An earlier version appeared as DEC Systems Research Center Research Report #45, February 1989.

[Car92] Luca Cardelli. Extensible records in a pure calculus of subtyping. Technical Report 81, Digital Equipment Corporation, System Research, 1992. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press, 1994).

[Car93] Luca Cardelli. An implementation of $F_{<:}$. Research report 97, DEC Systems Research Center, February 1993.

[Car95] Luca Cardelli. A language with distributed scope. *Computing Systems*, 8(1):27–59, 1995. Short version in *Proceedings of POPL '95.* A preliminary version appeared as Report 122, Digital Systems Research, June 1994.

[Car97] Luca Cardelli. Type systems. In Allen B. Tucker, editor, *Handbook of Computer Science and Engineering.*, chapter 103. CRC Press, 1997.

[Cas93] Guiseppe Castagna. A meta-language for typed object-oriented languages. In R. K. Shyamasundar, editor, *Foundations of Software Technology and Theoretical Computer Science*, volume 761 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1993.

[Cas95] Guiseppe Castagna. $\mathrm{F}^{\&}_{\leq}$: Integrating parametric and "ad-hoc" second order polymorphism. *Formal Aspects of Computing*, 8(3):247–293, 1995. A preliminary version appeared in the Proceedings of the 4th International Workshop on Database Programming Languages, 1994 (p. 338-358).

[Cas97] Guiseppe Castagna. *Object-Oriented Programming: A Unified Foundation.* Progress in Theoretical Computer Science. Birkhäuser, 1997.

[CC91] Felice Cardone and Mario Coppo. Type inference with recursive types: Syntax and semantics. *Information and Computation*, 92(1):48–80, 1991.

[CCF$^+$95] C. Cornes, J. Courant, J-C. Filiatre, G. Huet, P. Manoury, C. Mounoz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saïbi, and B. Werner. The Coq Proof Assistant User's Guide. Rapports Techniques 0177, INRIA Rocquencourt, Projet Formel, 1995. Version 5.10.

[CCHM89] Peter Canning, William Cook, Walt Hill, and Walter Olthoff John C. Mitchell. F-bounded polymorphism for object-oriented programming. In *Fourth ACM Conference on Functional Programming Languages and Computer Architecture*, Lecture Notes in Computer Science, pages 273–280. ACM, Springer, September 1989.

[CF58] Haskell B. Curry and Robert Feys. *Combinatory Logic.* North-Holland, 1958.

[CG90] Pierre-Louis Curien and Giorgio Ghelli. Coherence of subsumption. Technical Report LIENS 90-10, Laboratoire d'Informatique de l'Ecole Normale Supérieure, February 1990.

[CG92] Pierre-Louis Curien and Giorgio Ghelli. Coherence of subsumption: Minimum typing and type-checking in $F_{\leq}$. *Mathematical Structures in Computer Science*, 2:55–91, 1992. A preliminary version appeard as LIENS Report Nr. 90-10, 1990. Also in the collection [GM94].

[CG97] Adriana Compagnoni and Healfdene Goguen. Typed operational semantics for higher order subtyping. Technical Report ECS-LFCS-97-361, Department of Computer Science, University of Edinburgh, 1997. Submitted for publication in *Information and Computation.*

[CGL95] Guiseppe Castagna, Giorgio Ghelli, and Guiseppe Longo. A calculus for overloaded functions with subtyping. *Information and Computation*, 117(1):115–135, February 1995. A Preliminary Version appeared in ACM Conference on LISP and Functional Programming, June 1992 (pp. 182–192) and as LIENS Rapport de Recherche, LIENS-92-4, 1992.

[Cha92] Craig Chambers. Object-oriented multi-methods in Cecil. In O. Lehrmann Madsen, editor, *ECOOP '92*, volume 615 of *Lecture Notes in Computer Science*, pages 33–56. Springer, 1992.

[Cha93] Craig Chambers. The Cecil language: Specification and rationale. Technical Report 93-03-05, Department of Computater Science and Engineering. University of Washington, 1993.

[CHC90] William R. Cook, Walter L. Hill, and Peter S. Canning. Inheritance is not subtyping. In *Seventeenth Annual Symposium on Principles of Programming Languages (POPL) (San Fancisco, CA)* [ACM90], pages 125–135. Also in the collection [GM94].

[Che96] Gang Chen. Subtyping calculus of constructions (extended abstact). Technical report, DMI-LIENS, December 1996.

[Che97a] Gang Chen. Dependent type systems with subtyping; type level transitivity elimination. Technical report, Laboratoire d'Informatique ENS. and Université de Paris 7, July 1997. To appear in the Proceedings of the KIT 97 Summer School and Workshop, Beijing.

[Che97b] Gang Chen. Subtyping calculus of constructions (extended abstract). In *Proceedings of Mathematical Foundations of Computer Science (MFCS '97)*, 1997. A longer version is available through `http://www.dmi.ens.fr/~gang`.

[Chu40] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(1):56–68, 1940.

[CL96] Gang Chen and Guiseppe Longo. Subtyping parametric and dependent types, an introduction. Technical report, DMI-LIENS, December 1996.

[CM91] Luca Cardelli and John Mitchell. Operations on records. *Mathematical Structures in Computer Science*, 1:3–48, 1991. Also in the collection [GM94]; available as DEC Systems Research Center Research Report #48, August, 1989, and in the proceedings of MFPS '89, Springer LNCS volume 442.

[CMMŠ94] Luca Cardelli, Simone Martini, John Mitchell, and André Ščedrov. An extension of system F with subtyping. *Information and Computation*, 109(1–2):4–56, 1994. A preliminary version appeared in TACS '91 (Sendai, Japan, pp. 750–770, LNCS 526).

[Com95a] Adriana B. Compagnoni. Decidability of higher-order subtyping with intersection types. In Pacholski and Tiuryn [PT95a]. Also available as University of Edinburgh, LFCS technical report ECS-LFCS-94-281, titled "Subtyping in $F^{\omega}_{\wedge}$ is decidable".

[Com95b] Adriana B. Compagnoni. *Higher-Order Subtyping with Intersection Types*. PhD thesis, Catholic University, Nijmegen, January 1995.

[Com97a] Adriana Compagnoni. Decidable higher-order subtyping. Technical report, Laboratory for Foundations of Computer Science, University of Edinburgh, August 1997. Draft Techreport.

[Com97b] Adriana Compagnoni. Subject reduction and minimal types for higher order subtyping. Technical report, Laboratory for Foundations of Computer Science, University of Edinburgh, August 1997. Draft Techreport.

[Coo87] William Cook. A *self*-ish model of inheritance. Manuscript, 1987.

[Coo89] William Cook. *A Denotational Model of Inheritance.* PhD thesis, Brown University, 1989.

[Coo91] William Cook. Object-Oriented Programming Versus Abtract Data Types. In de Bakker et al. [dBdR91], pages 151–178.

[CP89] W. Cook and J. Palsberg. A denotational semantics of inheritance and its correctness. In *Object Oriented Programing: Systems, Languages, and Applications (OOPSLA) '89 (New Orleans, LA)*, pages 433–444. ACM, 1989. in *SIGPLAN Notices* 24(10).

[CP96] Adriana B. Compagnoni and Benjamin C. Pierce. Intersection types and multiple inheritance. *Mathematical Structures in Computer Science*, 6(5):469–501, October 1996. Preliminary version available as University of Edinburgh technical report ECS-LFCS-93-275 and Catholic University Nijmegen computer science technical report 93-18, Aug. 1993, under the title *Multiple Inheritance via Intersection Types.*

[Cra96a] Karl Crary. *KML Reference Manual.* Department of Computer Science, Cornell University, 1996.

[Cra96b] Karl Crary. A unified framework for modules and objects and its application to programming language design (extended abstract). Technical report, Department of Computer Science, Cornell University, September 1996.

[Cra97] Karl Crary. Foundations for the implementation of higher-order subtyping. In *Proceedings of the 2nd International Conference on Functional Programming (IFCP' 97), Amsterdam.* ACM Press, June 1997.

[CW85] Luca Cardelli and Peter Wegner. On understanding types, data abstraction and polymorphism. *Computing Surveys*, 17(4):471–522, 1985.

[dBdR91] J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors. *Foundations of Object-Oriented Languages (REX Workshop)*, volume 489 of *Lecture Notes in Computer Science*. Springer, 1991.

[DF96] Paolo Diblasio and Kathleen Fisher. A concurrent object calculus. In Ugo Montanari and Vladimiro Sassone, editors, *Proceedings of CONCUR '96*, volume 1119 of *Lecture Notes in Computer Science*, pages 655–670. Springer, 1996. An extended version appeared as Stanford University Technical Note STAN-CS-TN-96-36, 1996.

[DG87] L. G. DeMichiel and R. P. Gabriel. Common Lisp Object System oberview. In J. Bézivin, J.-M. Hullot, P. Cointe, and H. Lieberman, editors, *ECOOP '87*, volume 276 of *Lecture Notes in Computer Science*, pages 151–170. Springer, 1987.

[DGLM95] Mark Day, Robert Gruber, Barbara Liskov, and Andrew C. Myers. subtypes vs. where clauses: Constraining parametric polymorphism. In *Object Oriented Programing: Systems, Languages, and Applications (OOPSLA) '95*, pages 156–158. ACM, 1995. in *SIGPLAN Notices*) 30(10).

[DM82] Luis Damas and Robin Milner. Principal type-schemes for functional programming languages. In *Ninth Annual Symposium on Principles of Programming Languages (POPL) (Albuquerque, NM)*, pages 207–212. ACM, January 1982.

[ES90] M. Ellis and B. Stroustrup. *The Annotated C++ Reference Manual.* Addison-Wesley, 1990.

[Fis96] Kathleen Fisher. *Type Systems for Object-Oriented Languages.* PhD thesis, Stanford University, August 1996.

[FM95] Kathleen Fisher and John Mitchell. A delegation-based object calculus with subtyping. In Horst Reichel, editor, *Proceedings of FCT '95*, volume 965 of *Lecture Notes in Computer Science*. Springer, 1995.

[FM96] Kathleen Fisher and John Mitchell. The development of type systems for object-oriented languages. *Theory and Practice of Object Systems*, 1(3):189–220, 1996. An earlier version appeared in [HM94] under the title “Notes on Typed Object-Oriented Programming”.

[FMRS92] P. Freyd, P. Mulry, G. Rosolini, and D. Scott. Extensional PERs. *Information and Computation*, pages 211–227, 1992. A preliminary version appeared in Proceedings of LICS 1990, pages 346-354.

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

[Ghe91] Giorgio Ghelli. Modelling features of object-oriented languages in second order functional languages with subtypes. In de Bakker et al. [dBdR91], pages 311–340.

[Ghe93] Giorgio Ghelli. Recursive types are not conservative over $F_{\leq}$. In M. Bezem and J.F. Groote, editors, *Typed Lambda Calculus and Applications*, volume 664 of *Lecture Notes in Computer Science*, March 16-18, 1993, Utrecht, The Netherlands, 1993. Springer.

[Ghe95] Giorgio Ghelli. Divergence of $F_{\leq}$ type checking. *Theoretical Computer Science*, 139(1,2):131–162, 1995. Also appeared as Dipartimento di Informatica, Universitá di Pisa, Technical Report 5/93, 1993.

[Gir71] Jean-Yves Girard. Une extension de l'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types. In J. E. Fenstad, editor, *Second Scandinavian Logic Symposium '71 (Oslo, Norway)*, number 63 in Studies in Logic and the Foundations of Mathematics, pages 63–92. North-Holland, 1971.

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

[GJ96] Benedict R. Gaster and Mark P. Jones. A polymorphic type system for extensible records and variants. Technical Report NOTTCS-TR-96-3,, Functional Programming Research Group, Department of Computer Science, University of Nottingham, 1996.

[GLT89] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types.* Cambridge University Press, 1989.

[GM94] Carl A. Gunter and John C. Mitchell. *Theoretical Aspects of Object-Oriented Programming, Types, Semantics, and Language Design.* Foundations of Computing Series. MIT Press, 1994.

[Gog94a] Healfdene Goguen. The metatheory of UTT. In P P. Dybjer, B. Nordstroem, and J. Smith, editors, *Types for Proofs and Programs, International Workshop (TYPES '94); Bastad, Sweden*, volume 996 of *Lecture Notes in Computer Science*, pages 60–82. Springer, June 1994.

[Gog94b] Healfdene Goguen. *Typed Operational Semantics for Type Theory.* PhD thesis, University of Edinburgh, 1994.

[Gog95] Healfdene Goguen. Typed operational semantics. In Mariangiola Dezani-Ciancaglini and Gordon Plotkin, editors, *Second International Conference on Typed Lambda Calculi and Applications (TCLA '95)*, volume 902 of *Lecture Notes in Computer Science*, pages 186–200. Springer, 1995.

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

[GR83] Adele Goldberg and David Robson. *Smalltalk-80: The Language and its Implemementation.* Addison-Wesley, Reading, MA, 1983.

[Gun92] Carl A. Gunter. *Semantics of Programming Languages.* MIT Press, 1992.

[Hay94] Susumu Hayashi. Singleton, union and intersection types for program extraction. *Information and Computation*, 1994.

[Hen94] Andreas V. Hense. *Polymorphic Type Inference for Object-Oriented Programming Languages.* PhD thesis, Technische Fakultät der Universität des Saarlandes, 1994.

[Hic96] Jason J. Hickey. Nuprl-Light: An implementation framework for higher-order logic. Technical report, Cornell University, 1996. Available electronically through `http://www.cs.cornell.edu/home/jyh`.

[Hic97] Jason J. Hickey. A semantics of objects in type theory (forthcoming). Technical report, Computer Science Department, Cornell University, 1997.

[Hin69] J. R. Hindley. The principal type-scheme of an object in combinatory logic. *Transactions of the American Mathematical Society*, 146:29–60, December 1969.

[HM94] M. Hagiya and John C. Mitchell, editors. *Theoretical Aspects of Computer Software*, volume 789 of *Lecture Notes in Computer Science.* Springer, 1994.

[HNSS98] Martin Hofmann, Wolfgang Naraschewski, Martin Steffen, and Terry Stroup. Inheritance of proofs. *Theory and Practice of Object Systems (Tapos), Special Issue on Third Workshop on Foundations of Object-Oriented Languages (FOOL 3), July 1996*, 4(1):51–69, January 1998. An extended version appeared as Interner Bericht, Universität Erlangen-Nürnberg, IMMDVII-5/96.

[How80] W. A. Howard. The formulae-as-types-notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[HP91] R. Harper and B. Pierce. A record calculus based on symmetric concatenation. In *Eighteenth Annual Symposium on Principles of Programming Languages (POPL) (Orlando, FL)* [ACM91], pages 131–142. Extended version available as Carnegie Mellon Technical Report CMU-CS-90-157.

[HP95a] Martin Hofmann and Benjamin Pierce. Positive subtyping. In *22nd Annual Symposium on Principles of Programming Languages (POPL) (San Francisco, California)*, pages 186–197. ACM, January 1995. Full version in *Information and Computation*, volume 126, number 1, April 1996. Also available as University of Edinburgh technical report ECS-LFCS-94-303, September 1994.

[HP95b] Martin Hofmann and Benjamin Pierce. A unifying type-theoretic framework for objects. *Journal of Functional Programming*, 5(4):593–635, October 1995. Previous versions appeared in the Symposium on Theoretical Aspects of Computer Science, 1994, (pages 251–262) and, under the title "An Abstract View of Objects and Subtyping (Preliminary Report)," as University of Edinburgh, LFCS technical report ECS-LFCS-92-226, 1992.

[HT91] Kohei Honda and Mario Tokoro. An object calculus for asynchronous communication. In P. America, editor, *ECOOP '91*, volume 512 of *Lecture Notes in Computer Science*, pages 133–147. Springer, 1991.

[HT92] Kohei Honda and Mario Tokoro. On asynchronous communication semantics. In Tokoro et al. [TNW92], pages 21–51.

[IEE90] IEEE. *Fifth Annual Symposium on Logic in Computer Science (LICS)*. Computer Society Press, June 1990.

[JM93] L. Jategaonkar and J. Mitchell. Type inference with extended pattern matching and subtypes. *Acta Informatica*, 19:127–166, 1993. A preliminary version appeared in [ACM88a] (p. 198-211) under the title "ML with extended pattern matching and subtypes".

[Jon93] Cliff Jones. A pi-calculus semantics for an object-based design notation. In Eike Best, editor, *Proceedings of CONCUR '93*, volume 715 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1993.

[Jon97] M. P. Jones. First-class polymorphism with type inference. In POPL '97 [POP97].

[KG89] Sonya Keene and Dan Gerson. *Object-Oriented Programming in Common Lisp*. Addison-Wesley, Reading, Massachusetts, 1989.

[KLMM94] Dinesh Katiyar, David Luckham, S. Meldal, and John Mitchell. Polymorphism and subtyping in interfaces. In *Workshop on Interface Definition Languages*, pages 22–34. ACM, 1994. In *SIGPLAN Notices*) 29(8).

[KMP84] Gilles Kahn, David MacQueen, and Gordon Plotkin, editors. *Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science*. Springer, 1984.

[KR94] Samuel N. Kamin and Uday S. Reddy. Two semantic models of object-oriented languages. In *Theoretical Aspects of Object-Oriented Programming, Types, Semantics, and Language Design* [GM94], pages 464–495.

[KY94] Naoki Kobayashi and Akinori Yonezawa. Type-theoretic foundations for concurrent object-oriented programming. In *Object Oriented Programing: Systems, Languages, and Applications (OOPSLA) '94*. ACM, 1994. to appear.

[LCD$^+$94] Barbara Liskov, Dorothy Curtis, Mark Day, Snjay Ghemawat, Robert Gruber, Paul Johnson, and Andrew C. Myers. *Theta Reference Manual*. MIT Laboratory for Computer Science, Cambridge, Massachusetts, February 1994. Available at http://www.pgm.lcs.mit.edu/papers/thetaref/.

[leg97] The Lego World Wide Web page, 1997. Accessible electronically through `http://www.dcs.ed.ac.uk/home/lego`.

[Liq97] Luigi Liquori. An extended theory of primitive objects: First order systems. In Akşit and Matsuoka [AM97].

[LP91] Wilf LaLonde and John Pugh. Subclassing $\neq$ subtyping $\neq$ *is-a*. *Journal of Object-Oriented Programming*, pages 57–62, January 1991.

[LPT89] Zhaohui Luo, Randy Pollack, and Paul Taylor. How to use LEGO (a preliminary user's manual). Technical Report 27, Laboratory for Foundations of Computer Science, University of Edinburgh, October 1989.

[Luo92] Zhaohui Luo. A unifying theory of dependent types: the schematic approach. Technical Report ECS-LFCS-92-202, Laboratory for Foundations of Computer Science, University of Edinburgh, March 1992.

[Luo95] Zhaohui Luo. Developing reuse technology in proof engineering. In *Proceedings of the AISB '95 Workshop on Automated Reasoning, Sheffield, UK*, 1995.

[Luo96] Zhaohui Luo. Coercive subtyping in type theory. In *Computer Science Logic (CSL '96)*, volume xxx of *Lecture Notes in Computer Science.* Springer, 1996.

[LY96] T. Lindholm and F. Yellin. *The Java Virtual Machine.* Addison-Wesley, 1996.

[Ma92] QuingMing Ma. Parametricity as subtyping. In *Nineteenth Annual Symposium on Principles of Programming Languages (POPL) (Albuquerque, NM)* [ACM92], pages 281–292.

[MBL97] Andrew C. Myers, Joseph A. Bank, and Barbara Liskov. Parameterized types and Java. In POPL '97 [POP97], pages 132–145.

[Mey92] Bertrand Meyer. *Eiffel: The Language.* Prentice Hall, 1992.

[Mil78] R. Milner. A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17(3):348–375, December 1978.

[Mit90a] John Mitchell. Type systems for programming languages. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, chapter 8, pages 365–458. Elsevier, 1990.

[Mit90b] John C. Mitchell. Toward a typed foundation for method specialization and inheritance. In *Seventeenth Annual Symposium on Principles of Programming Languages (POPL) (San Fancisco, CA)* [ACM90], pages 109–124. Also in the collection [GM94].

[Mit96] John C. Mitchell. *Foundations of Programming Languages.* Foundation of Computing Series. MIT Press, 1996.

[ML75] Per Martin-Löf. An intuitionistic theory of types: Predicative part. In H. E. Rose and John C. Shepherdson, editors, *ASL Logic Colloquium '73 (Bristol, England)*, number 80 in Studies in Logic and the Foundations of Mathematics, pages 73–118. North-Holland, 1975.

[MP88] John C. Mitchell and Gordon D. Plotkin. Abstract types have existential type. *ACM Transactions on Programming Languages and Systems*, 10(3):470–502, July 1988.

[MPS86] David MacQueen, Gordon Plotkin, and Ravi Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71:95–130, 1986.

[MPW92] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, part I/II. *Information and Computation*, 100:1–77, 1992.

[MV96] John C. Mitchell and Ramesh Viswanathan. Effective models of polymorphism, subtyping, and recursion. In Friedrich Meyer auf der Heide and Burkhard Monien, editors, *23rd Colloquium on Automata, Languages and Programming (ICALP) (Paderborn, Germany)*, volume 1099 of *Lecture Notes in Computer Science*. Springer, 1996.

[Nie92] Oscar Nierstrasz. Towards an object calculus. In Tokoro et al. [TNW92], pages 1–20.

[OL92] Stephen M. Omohundro and Chu-Cheow Lim. The Sather languange and libraries. Tr-92–17, International Computer Science Institute, Berkeley, March 1992.

[Olt95] W. Olthoff, editor. *ECOOP '95*, volume 952 of *Lecture Notes in Computer Science*, 1995.

[OW97a] Martin Odersky and Philip Wadler. The Pizza compiler. Available through `http://www.math.luc.edu:/pizza`, 1997.

[OW97b] Martin Odersky and Philip Wadler. Pizza into Java: Translating theory into practice. In POPL '97 [POP97]. A preliminary version appeared as Technical Report, 26/96, University of Karlsruhe, July 1996.

[Pau93] Lawrence C. Paulson. The Isabelle reference manual. Technical Report 283, University of Cambridge, Computer Laboratory, 1993.

[PDM89] Benjamin Pierce, Scott Dietzen, and Spiro Michaylov. Programming in higher-order typed lambda-calculi. Technical Report CMU-CS-89-111, Carnegie Mellon University, March 1989.

[Pfe93a] Frank Pfenning. On the undecidability of partial polymorphic type reconstruction. *Acta Informatica*, 19(1,2):185–199, 1993.

[Pfe93b] Frank Pfenning. Refinement types for logical frameworks. In *Informal Proceedings of the 1993 Workshop on Types for Proofs and Programs*, pages 315–328, May 1993.

[Pho90] Wesley Phoa. Effective domains and intrinsic structure. In *Fifth Annual Symposium on Logic in Computer Science (LICS) (Philadelphia, PA)* [IEE90], pages 366–377.

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

[Pie97a] Benjamin C. Pierce. Bounded quantification with bottom. Technical Report CSCI TR #492, Indiana University, November 1997.

[Pie97b] Benjamin C. Pierce. Intersection types and bounded polymorphism. *Mathematical Structures in Computer Science*, 7(2):129–193, April 1997. Summary version appeared in Conference on Typed Lambda Calculi and Applications, March 1993, pp. 346–360. Preliminary version available as University of Edinburgh technical report ECS-LFCS-92-200.

[POP97] ACM. *24th Annual Symposium on Principles of Programming Languages (POPL) (Paris, France)*, January 1997.

[PP92] Benjamin Pierce and Randy Pollack. Higher-order subtyping. Unpublished Manuscript. In the Dienstagsclub's Blechschrank., November 1992.

[PS94] Jens Palsberg and Michael I. Schwartzbach. *Object-Oriented Type Systems*. John Wiley & Sons, 1994.

[PS96] Benjamin C. Pierce and Davide Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5):409–454, 1996.

[PS97] Benjamin Pierce and Martin Steffen. Higher-order subtyping. *Theoretical Computer Science*, 176(1,2):235–282, 1997. A shorter version appeared in the Proceedings IFIP Working Conference on Programming Concepts, Methods and Calculi (p. 511–530), 1994. Also LFCS technical report ECS-LFCS-94-280 and Interner Bericht IMMD7-01/94, Universität Erlangen.

[PT93] Benjamin Pierce and David Turner. Statically typed friendly functions via partially abstract types. Technical Report ECS-LFCS-93-256, Laboratory for Foundations of Computer Science, University of Edinburgh, 1993. Also available as INRIA-Rocquencourt Rapport de Recherche No. 1899.

[PT94] Benjamin Pierce and David Turner. Simple type-theoretic foundations for object-oriented programming. *Journal of Functional Programming*, 4(2):207–247, April 1994. A preliminary version appeared in Principles of Programming Languages, 1993, and as University of Edinburgh technical report ECS-LFCS-92-225, under the title "Object-Oriented Programming Without Recursive Types".

[PT95a] L. Pacholski and J. Tiuryn, editors. *Computer Science Logic (CSL '94)*, volume 933 of *Lecture Notes in Computer Science*. Springer, 1995.

[PT95b] Benjamin C. Pierce and David N. Turner. The Pict manual. Distributed with the Pict implementation, 1995.

[PT97a] Benjamin C. Pierce and David N. Turner. Local type argument synthesis with bounded quantification. Technical Report CSCI TR #495, Indiana University, November 1997.

[PT97b] Benjamin C. Pierce and David N. Turner. Pict: A programming language based on the pi-calculus. Technical Report CSCI 476, Computer Science Department, Indiana University, 1997. To appear in Milner *Festschrift*, MIT Press, 1997.

[PT97c] Benjamin C. Pierce and David N. Turner. Pict language definition (version 4.0). Available electronically through `http://www.cs.indiana.edu/hyplan/pierce/ftp/`, 1997.

[PT98] Benjamin C. Pierce and David N. Turner. Local type inference. In *Proceedings of POPL '98*. ACM, 1998. Also as Indiana University Technical Report CSCI TR #493.

[Red88] Uday S. Reddy. Objects as closures: Abstract semantics of object-oriented languages. In *Symposium on* LISP *and Functional Programming (Snowbird, UT)*, pages 289–297. ACM, July 1988.

[Rém89] Didier Rémy. Typechecking records and variants in a natural extension of ML. In *Sixteenth Annual Symposium on Principles of Programming Languages (POPL) (Austin, TX)*, pages 77–87. ACM, January 1989.

[Rém92] Didier Rémy. Typing record concatenation for free. In *Nineteenth Annual Symposium on Principles of Programming Languages (POPL) (Albuquerque, NM)* [ACM92], pages 166–176. Also in Carl A. Gunter and John C. Mitchell, editors, *Theoretical Aspects of Object-Oriented Programming: Types, Semantics, and Language Design* (MIT Press, 1994).

[Rey74] John Reynolds. Towards a theory of type structure. In B. Robinet, editor, *Colloque sur la programmation (Paris, France)*, volume 19 of *Lecture Notes in Computer Science*, pages 408–425. Springer, 1974.

[Rey85] John C. Reynolds. Three approaches to type structure. In H. Ehrig, C. Floyd, M. Nivat, and J. Thatcher, editors, *Mathematical Foundations of Software Development, Volume1: Colloquium on Trees in Algebra and Programming (CAAP '85)*, volume 185 of *Lecture Notes in Computer Science*, pages 97–138. Springer, 1985.

[Rey96] John C. Reynolds. Design of the programming language Forsythe. Technical Report CMU-CS-96-146, Carnegie Mellon University, June 1996. A preliminary version appeared as technical report CMU-CS-88-159.

[RIR93] N. Rodriguez, R. Ierusalimschy, and J. L. Rangel. Types in School. *ACM SIGPLAN Notices*, 28, 8 1993.

[SCB$^+$86] Craig Schaffert, Topher Cooper, Bruce Bullis, Mike Kilian, and Carrie Wilpolt. An introduction to Trellis/OWL. In *Object Oriented Programing: Systems, Languages, and Applications (OOPSLA) '86 (Portland, Oregon)* [ACM86], pages 9–16. Special issue of SIGPLAN Notices (vol. 21 No. 11, November, 1986), a preliminary version appeared as Digital Equipment Technical Report, DEC-TR-372, November 1985.

[Sny86] A. Snyder. Encapsulation and inheritance in object-oriented programming languages. In *Object Oriented Programing: Systems, Languages, and Applications (OOPSLA) '86 (Portland, Oregon)* [ACM86], pages 38–45. in *SIGPLAN Notices* 21(11).

[SP94] Martin Steffen and Benjamin Pierce. Higher-order subtyping. Interner Bericht IMMD7-01/94, Informatik VII, Universität Erlangen-Nürnberg, January 1994. Also as LFCS technical report ECS-LFCS-94-280, accepted for publication in Theoretical Computer Science.

[Sta88] R. Stansifer. Type inference with subtypes. In *Fifteenth Annual Symposium on Principles of Programming Languages (POPL) (San Diego, CA)* [ACM88b], pages 88–97.

[Str86] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley, 1986.

[Tai67] William W. Tait. Intensional interpretation of functionals of finite type. *Journal of Symbolic Logic*, 32, 1967.

[Tai75] William W. Tait. A realizability interpretation of the theory of species. In R. Parikh, editor, *Logic Colloquium '75*, number 453 in Lecture Notes in Mathematics. Springer, 1975.

[Tho91] Simon Thompson. *Type Theory and Functional Programming.* International Computer Science Series. Addison-Wesley, 1991.

[TNW92] M. Tokoro, O. Nierstrasz, and P. Wegner, editors. *Object-Based Concurrent Computing 1991*, volume 612 of *Lecture Notes in Computer Science.* Springer, 1992.

[US91] D. Ungar and R.B. Smith. Self: The power of simplicity. *Lisp and Symbolic Computation*, 4(3):187–206, 1991. Preliminary version appeared in *Proc. ACM Symp. on Object-Oriented Programming: Systems, Languages, and Applications,* 1987, 227-241.

[Vas94] Vasco Vasconcelos. Typed concurrent objects. In M. Tokoro and R. Pareschi, editors, *Proceedings of ECOOP '94*, volume 821 of *Lecture Notes in Computer Science*, pages 100–117. Springer, 1994.

[VT93] Vasco Thudichum Vasconcelos and Mario Tokoro. A typing system for a calculus of objects. In Shojiro Nishio and Akinori Yonezawa, editors, *First JSSST International Symposion on Object Technologies for Advances Software (Kanazawa, Japan)*, volume 742 of *Lecture Notes in Computer Science*, pages 460–474. Springer, 1993.

[Wan87a] Mitchell Wand. Complete type inference for simple objects. In *Second Annual Symposium on Logic in Computer Science (LICS) (Ithaca, NY)*, pages 37–44. IEEE, Computer Society Press, June 1987.

[Wan87b] Mitchell Wand. A simple algorithm and proof for type inference. *Fundamenta Informaticae*, 10:115–122, 1987.

[Wan88] Mitchell Wand. Corrigendum: Complete type inference for simple objects. In *Third Annual Symposium on Logic in Computer Science (LICS) (Edinburgh, Scotland).* IEEE, Computer Society Press, June 1988.

[Wan91] Mitchell Wand. Type inference for record concatenation and multiple inheritance. *Information and Computation*, 92:1–15, 1991. A preliminary version appeared in the Proceedings of LICS'89, pages 92-97.

[Wan94] Mitchell Wand. Type inference for objects with instance variables and inheritance. In *Theoretical Aspects of Object-Oriented Programming, Types, Semantics, and Language Design* [GM94], pages 97–120. Also appeared as Northeastern University Technical Report, NU-CCS-89-2, 1989.

[Wel94] J. B. Wells. Typability and type checking in the second order λ-calculus are equivalent and undecidable. In *Ninth Annual Symposium on Logic in Computer Science (LICS) (Paris, France)*, pages 176–185. IEEE, Computer Society Press, July 1994.

[Wra89] Gavin C. Wraith. A note on categorical datatypes. In David Pitt, David Rydeheard, Peter Dybjer, Andrew Pitts, and Axel Poigné, editors, *Category Theory and Computer Science (Manchester, U.K.)*, volume 389 of *Lecture Notes in Computer Science*, pages 118–127. Springer, September 1989.

[Yon90] Akinori Yonezawa. *ABCL: An Object-Oriented Concurrent System.* MIT Press, 1990.

[Zwa96] Jan Zwanenburg. A type system for record concatenation and subtyping. In Bruce [Bru96a]. Available electronically through `http://www.cs.williams.edu/~kim/FOOL/Abstracts.html`.

Part IV

German translations

Polarisierte Untertypen höherer Ordnung

Der Technischen Fakultät der
Universität Erlangen-Nürnberg

zur Erlangung des Grades

D O K T O R – I N G E N I E U R

vorgelegt von

Martin Steffen

Erlangen – 1997

Zusammenfassung

Der Kalkül $F^{\omega}_{\leq}$, ein polymorpher λ-Kalkül mit Untertypen und Typoperatoren, wurde als Basiskalkül für typisierte, objektorientierte Sprachen vorgeschlagen. Die in der Literatur untersuchten Versionen unterstützen in der Regel nur punktweise Untertypisierung von Typoperatoren, das heißt, zwei Applikationen $S\ U$ und $T\ U$ stehen in Untertypbeziehung, wenn S und T dies tun. Cardelli präsentiert in seiner häufig zitierten, unveröffentlichten Note $F^{\omega}_{\leq}$ in einer allgemeineren Form. Sie geht über die punktweise Untertypisierung von Applikationen hinaus, indem sie zwischen monotonen und antimonotonen Operatoren unterscheidet. Für einen monotonen Operator T, beispielsweise, impliziert $U_1 \leq U_2$, daß auch $T\ U_1$ ein Untertyp von $T\ U_2$ ist.

Meine These erweitert $F^{\omega}_{\leq}$ um polarisierte Applikation, entwickelt seine Metatheorie, und weist die Entscheidbarkeit des polarisierten $F^{\omega}_{\leq}$ nach. Die Einführung polarisierter Applikationen führt zu wechselseitiger Abhängigkeit des Untertyp– und des Artsystems. Dies steht im Gegensatz zum unpolarisierten Fall, bei dem Untertypableitungen von Artableitungen abhängen, aber nicht umgekehrt. Um die Entscheidbarkeit des System nicht zu verlieren, stellt die Arbeit eine Verallgemeinerung der bekannten Untertypregel für All-Typen vor, die auf der Identität der Oberschranken der verglichenen Alltypen beharrt. Die Verallgemeinerung im polarisierten Fall besteht darin, daß dies zu einer Forderung nach gegenseitiger Untertypbeziehung der Oberschranken abgeschwächt wird.

Kapitel 1

Einleitung

Das einleitende Kapitel dient der Begriffsklärung und dazu, relevante Forschungsergebnisse auf dem Gebiet zusammenzutragen. Schließlich gebe ich Hinweise auf weitergehende Literatur und einen Überblick über den Inhalt meiner These.

1.1 Typen und Programme

Typen sind ein natürliches, programmiersprachliches Konzept. Der Typ eines Programmes beschreibt, in welchen Kontexten es verwendet werden darf. Unter starker Typisierung versteht man, daß Laufzeitfehler durch Fehlverwendung des Programmes, beispielsweise die Anwendung einer Funktion auf ein unpassendes Argument oder ein Aufruf einer nicht unterstützten Methode an einem Objekt, ausgeschlossen sind. Eine Sprache ist typsicher, wenn derartige Fehler durch ein Typsystem abgefangen werden. Statisch typisierte Sprachen überprüfen das Wohlverhalten ihrer Programmes bevor das Programm ausgeführt wird, zur Übersetzungszeit.

Das Typsystem einer Programmiersprache bildet ein formales Deduktionssystem und unterteilt die Mengen der syntaktisch korrekten Programme in wohltypisierte und nicht-wohltypisierte. Dies trägt zum Schutz gegen nicht-systematische Programmierfehler und somit zu zuverlässigerer Software bei. Daneben erhöht die statische Typisierung der Programme die Effizienz, indem sie das Laufzeitsystem von kostspieligen Überprüfungen des laufenden Programmes befreit. Indem die Typen die beabsichtigte Verwendung eines Programmes dokumentieren, erhöhen sie die Lesbarkeit des Quellkodes. Dies trifft vor allem für große, strukturierte Programme, die in Objekte oder Module unterteilt sind, zu, deren Schnittstellen durch ihre Typen spezifiziert werden. Typüberprüfung bedeutet hier den Nachweis der Konsistenz der Schnittstellen. Schließlich benutzen Übersetzer Typinformation, um effizienteren Kode zu generieren.

Nicht alle möglichen Laufzeitfehler sind statisch überprüfbar, weil die Eigenschaft unentscheidbar sein mag oder weil der Test schlicht zu aufwendig wäre. Da ein sicheres, statisches Typsystem, was die akzeptierten Programme betrifft, konservativ sein muß, wird es Programme zurückweisen, die ansonsten fehlerfrei liefen, und je schwächer das Typsystem, desto mehr Programme wird es ablehnen. Der Entwurf starker, doch sicherer und entscheidbarer Typsysteme ist ein Hauptthema in der Evolution der Programmiersprachen und ein wichtiger Beitrag der theoretischen Informatik. Für objektorientierte Sprachen ist die Entwicklung beweisbar sicherer, ausdrucksstarker Typsysteme eine besondere Herausforderung, da diese Sprachen für ihre Vielfalt an unterschiedlichen, mächtigen — doch nicht notwendigerweise wohlverstandenen — Konzepten bekannt sind.

1.2 Objektorientierte Programmiersprachen

Objekte und Klassen

Selbst wenn keine Einmütigkeit darüber besteht, durch welche Merkmalskombination genau sich objektorientierte Sprachen auszeichnen sollten, eines ist unbestritten: Programme sind in *Objekte* gegliedert. Ein Objekt enthält einen internen *Zustand* und Prozeduren, die *Methoden* des Objektes, um auf ihn zugreifen zu können. Die Methoden stellen die Schnittstelle des Objektes dar und bieten die einzige Zugriffsmöglichkeit auf seinen Zustand. In diesem Sinne kapseln und abstrahieren Objekte von ihrem internen Zustand. Vor allem bei der Entwicklung großer Programme und Bibliotheken sind saubere Schnittstellen– und Abstraktionsmechanismen wichtig. ([Coo91] diskutiert die Unterschiede zwischen Abstrakten Datentypen und Objekten.)

Ein wesentlicher Zug objektorientierter Sprachen ist die *dynamische Bindung* von Methoden, auch als späte Methodenbindung oder als dynamischer Methodenaufruf bezeichnet. Das selbe ist gemeint, spricht man davon, einem Objekt eine Nachricht zu senden. Wie immer man es bezeichnet, man versteht darunter, daß der Kode einer Methode nicht statisch zur Übersetzungzeit, bei der Konstruktion des Objektes, sondern beim Aufruf der Methode zur Laufzeit ausgewählt wird. Dies ist der Hauptunterschied zwischen Funktionsanwendung und Methodenaufruf. Da es möglich ist, Methoden zu definieren, deren Definition sich auf andere Methoden des selben Objektes beziehen können (zum Beispiel in Smalltalk mittels des Schlüsselwortes *self* oder durch *this* in C^{++}) betrifft eine Re-Implementierung einer referenzierten Methode (durch sogenanntes *method override*) auch die referierenden Methoden des selben Objektes. Es sollte einleuchten, daß diese dynamische Eigenschaft der Vererbung für die statische Typüberprüfung ein Problem darstellt.

Objekte sind die Struktureinheiten fertiger Programme, doch ob Objekte auch im Mittelpunkt der Programmentwicklung stehen, ist eine getrennte Frage. In klassenbasierten Sprachen wie Simula [BDMN79], Smalltalk [GR83], C^{++} [Str86] [ES90], Eiffel [Mey92], Java [AG96] [LY96], um einige zu nennen, werden zur inkrementellen Programmentwicklung *Klassen* verwendet.[1]

Klassen als Bauplan für Objekte, enthalten die Beschreibung ihrer Implementierung, also Anfangswerte für die internen Daten und den Kode ihrer Methoden. Klassen können auf zweierlei Weise verwendet werden: zu einen, um neuen Objekte zu generieren. Die Objekte, die zu einer Klasse gehören und die selbe Implementierung teilen, sind die *Instanzen* der Klasse. Zum zweiten, um durch *Vererbung* inkrementell neue Klassen zu definieren. Inkrementell, da zur Definition der neuen Klasse Teile der alten Oberklasse wiederverwendet, d.h. geerbt, werden können, alte Methoden können durch neue ersetzt und neue Methoden hinzugefügt werden. Vererbung ist also ein Mechanismus zu Programmkonstruktion, der durch Vererbung die Wiederverwendung von Kode von Oberklassen und Unterklassen unterstützt, und große Programme und Bibliotheken in einer Vererbungshierarchie strukturiert.

Polymorphie

Wie alle höheren Programmiersprachen, die diesen Namen verdienen, unterstützen objektorientierte Sprachen Polymorphie in der einen oder anderen Form. Im Gegensatz zu einem monomorphen Programm, erlaubt ein polymorphes Eingaben unterschiedlichen Typs, was die Flexibilität und Ausdrucksstärke der Programmiersprache erhöht. (Eine Diskussion verschiedener Formen von Polymorphie findet sich in Cardelli und Wegner [CW85].)

[1]Die Mehrzahl objektorientierter Sprachen ist klassenbasiert. Die Alternative sind objektbasierte Sprachen, wie beispielsweise Self [US91]. Sie kommen ohne Klassen aus, indem die Erzeugung neuer Objekte unmittelbar an bereits bestehenden Objekte durchgeführt wird. Dies wird als Klonen bezeichnet und tritt an die Stelle von Vererbung und Instantiierung.

Parametrische Polymorphie

Parametrische Polymorphie erlaubt Programme, welche uniform auf Eingaben unterschiedlichen Typs arbeiten, indem dem Programm der Typ als Parameter übergeben werden kann.

Ein Standardbeispiel hierfür ist die Funktion, die die Komponenten eines Wertepaares eines gegebenen Typs vertauscht. Der Typ selbst spielt für die Wirkung der Funktion keine Rolle und die Funktion arbeitet uniform für alle Typen. Anstelle also für jeden Typ T eine spezielle Funktion des Typs $(T \times T) \rightarrow (T \times T)$ zu schreiben, für den die Funktion im Rest des Programmes gebraucht wird, ist es offensichtlich besser, die Funktion ein für alle Mal zu definieren. Die Funktion bekommt dann den allquantifizierten Typ $\forall A.(A \times A) \rightarrow (A \times A)$, um anzuzeigen, daß sie die geforderte Funktionalität uniform für alle Typen besitzt (A ist dabei eine Typvariable).

Universelle Typen kennzeichnen Funktionen, deren erstes Argument der Typ ist auf dem sie arbeiten. Typabstraktion zur Definition polymorpher Funktionen und Typapplikation zu ihrer Instantiierung ist ein eleganter und mächtiger Mechanismus zur Wiederverwendung von Kode. Darüberhinaus hilft es dem Compiler, Kodeverdoppelung zu vermeiden. Für objektorientierte Sprachen ermöglicht parametrischer Polymorphie, sogenannte *type casts* zu vermeiden und zu effizienteren Kode zu erzeugen.

Diese Form des expliziten parametrischen Polymorphie steht im Gegensatz zu den polymorphen Typsystemen vieler moderner Sprachen wie beispielsweise ML, Haskell oder Miranda, deren Typ*inferenz*system dem Programmierer die explizite Angabe des Typparameters einer Funktion in den meisten Fällen erspart. Diese sogenannten Hindley-Milner Typsysteme [Hin69] [Mil78] [DM82] (vgl. auch [Car87]) oder Systeme mit „let-Polymorphie” sind schwächer, indem sie implizit von einer Pränexquantifizierung der Typvariablen in den Typen polymorpher Funktionen ausgehen. Der Vorteil dieser Einschränkung besteht darin, daß die Typinferenz, anders als im allgemeinen Fall uneingeschränkter Polymorphie höherer Ordnung [Boe85] [Pfe93a] [Wel94], entscheidbar ist.

Untertyppolymorphie Die Untertyprelation basiert auf einem einfachen Gedanken: Typen sind geordnet und Programme des kleineren Typs können anstelle von Programmen eines größeren Typs verwendet werden. Als ein nicht speziell objektorientiertes Beispiel, betrachte den Typ der ganzen Zahlen und der reellen Zahlen. Auch wenn das Typsystem die Verwendung reeller Zahlen an Stellen verhindern muß, an denen ein ganzzahliger Wert erwartet wird, so liegt aus Sicht der Programmierers nichts Verkehrtes darin, Integerwerte anstelle von reellen Werten zu verwenden, schließlich fällt jede ganze Zahl auch unter die reellen Zahlen. Mit anderen Worten, die Menge der reellen Zahlen *subsumiert* die Menge der ganzen Zahlen, man bezeichnet den Typ

Int als einen Untertyp von *Real* und schreibt $Int \leq Real$. Aus Sicht des Compilers werden sich die Elemente beider Typen allerdings in ihrer Darstellung unterscheiden und er wird beispielsweise die direkte Addition unterschiedlich dargestellter Zahlen durch Umwandlung in eine gemeinsame Fließkommadarstellung verhindern.[2]

Erlaubte eine Sprache mit Untertypen nichts weiter als die gelegentliche Verwendung ganzer Zahlen anstelle reeller, würde man noch kaum von einem Untertypsystem sprechen wollen. Üblicherweise umfaßt eine Programmiersprache neben Grundtypen wie beispielsweise den der ganzen und der reellen Zahlen auch zusammengesetzte Typen. Eine herausragende Rolle spielen insbesondere in funktionalen Sprachen die funktionalen Typen. Ein Pfeiltyp der Form $T_1 \rightarrow T_2$ beschreibt alle Funktionen mit Wertebereich T_1 und Zielbereich T_2. Entsprechend der grundlegenden Intuition für die Untertyprelation, daß nämlich die Werte des kleineren Typs gefahrlos und ohne Laufzeitfehler für Werte (hier Funktionen) des größeren Typ verwendet werden dürfen, müssen die Elemente der Untertypen des Pfeiltyps $T_1 \rightarrow T_2$ ebenfalls Funktionen sein. Genauer müssen die Funktionen eines Untertyps *mindestens* alle Werte aus T_1 als Eingabe verarbeiten können und andererseits *höchstens* Ergebnisse aus T_2 zurückgeben, wenn sie an die Stelle von Funktionen aus $T_1 \rightarrow T_2$ treten können sollen. Die folgende Inferenzregel drückt diese Beziehung aus:

$$\frac{T_1 \leq S_1 \qquad S_2 \leq T_2}{S_1 \rightarrow S_2 \leq T_1 \rightarrow T_2}$$

Da also die Untertyprelation für funktionale Typen ihre Richtung auf den Argumenttypen umkehrt, spricht man von kontravariantem Verhalten des Pfeiltyp in der Argumentposition und von kovariantem in der Werteposition.

Der Gedanke, die Untertyprelation über die Struktur ihren Typen zu definieren, wird als *strukturelle* Untertypisierung bezeichnet [Car88a] [Car88b] [Rey85] [Sta88].[3] Die Untertyprelation bildet im allgemeinen eine Präordnung auf der Menge aller Typen und ist induktiv über Deduktionsregeln, ähnlich der für Pfeiltypen, definiert. Die Gesamtheit dieser Regeln bildet das *Untertypsystem.*

Der Kern der Untertyprelation, daß Elemente eines kleineren Typs auch alle größeren Typen tragen und aus diesem Grund an die Stelle derer aller größeren Typen

[2]Allgemeiner bezeichnet man die nicht-uniforme Verwendung von Programmen als Überladen. In dem Beispiel ist nennt man die arithmetische Operation + überladen. Da sich die Additionsalgorithmen für ganze und reelle Zahlen unterscheiden, spricht man in diesem Fall auch von „ad-hoc“-Polymorphie, um es von den uniformen Definitionen des parametrischen Polymorphie zu unterscheiden.

[3]Dies steht im Gegensatz zu *deklarativer* Untertypisierung, bei der der Benutzer festlegen muß, welche Typen in Untertypbeziehung stehen. Daß daneben einige objektorientierte Sprachen wie beispielsweise C^{++} oder Eiffel, nicht zischen Klassen und Typen unterscheiden, trägt zur Verwirrung bei.

treten können, drückt die folgende Regel der *Subsumption* aus, die das Typsystem im engeren Sinne mit dem Untertypsystem verbindet:

$$\frac{t \in T' \qquad T' \leq T}{t \in T}$$

Untertyppolymorphie oder auch Inklusionspolymorphie ist ein Charakteristikum objektorientierter Sprachen. Beispielsweise kann ein Objekt mit mehr Methoden gefahrlos eines mit weniger Methoden ersetzen, ohne daß es zu Laufzeitfehlern wie "message-not-understood" kommt, mit anderen Worten, die Typen beider Objekte stehen in Untertypbeziehung.

Beschränkt-parametrische Polymorphie Ein wichtiger Beitrag für typisierte objektorientierte Sprachen ist die Verknüpfung von parametrischer oder universeller Polymorphie und Untertyppolymorphie. In ihrem einflußreichen Papier [CW85] schlugen Cardelli und Wegner einen polymorphen λ-Kalkül zweiter Ordnung mit beschränkt-parametrischer Polymorphie und Records als funktionales Modell für objektorientierte Sprachen vor. Der Kalkül ist unter dem Namen "Bounded Fun" bekannt. Nehmen wir als Standardbeispiel *Point* als den Typ aller Punkt-Objekte, die zum Lesen und Schreiben ihrer x-Koordinate die Methoden *getx* und *setx* bereitstellen. Der Methodenaufruf der *getx*-Methode derartiger Objekte kann nun mit dem beschränkt-universellen Typ $\forall A{\leq}Point.A \rightarrow Int$ typisiert werden, wobei die Typvariable A für alle Untertypen von *Point* steht. Die strukturelle Definition der Untertyprelation stellt sicher, daß die Elemente aller Untertypen von *Point* bestimmte Eigenschaften gemein haben; im Fall der Punkte wird jedes Objekt eines beliebigen Untertyps von *Point* zumindest die *getx*- und die *setx*-Methode unterstützen.

Das angedeutete Typsystem zweiter Ordnung erlaubt es nicht, komplexere Methoden wie das Überschreiben der x-Koordinate mittels *setx* mit $\forall A{\leq}Point.A \rightarrow Int \rightarrow A$ zu typisieren. Der Unterschied zwischen *getx* und *setx* besteht darin, daß die zweite Methode den Zustand verändert, während die erste ihn nur liest. Um derartige zustandsverändernde Methoden typisieren zu können, wurde verschiedene Erweiterungen der λ-Kalküls zweiter Ordnung mit Untertypen vorgeschlagen, beispielsweise die Erweiterung auf F-beschränkte Polymorphie, Match-beschränkte Polymorphie oder Polymorphie höherer Ordnung. Mittlerweile gibt es eine Reihe von Sprachen, die beschränkt-universelle Polymorphie unterstützen, unter anderem Pizza, Rapide, Trellis/OWL [SCB$^+$86], Eiffel,, polyTOIL ... Daneben ist der Verzicht auf parametrisch-beschränkte Polymorphie ist einer der wesentlichen Nachteile von Java. Es gibt demzufolge eine Reihe von Vorschlägen, Java entsprechend zu erweitern, beispielsweise in [AFM97], oder in Odersky und Wadlers Sprache Pizza [OW97b] [OW97a]. Alle genannte Vorschläge beruhen auf F-beschränkter Polymorphie. Weitere Arbeiten in dieser Richtung sind [MBL97] [BLM96].

Vererbung Vererbung gestattet es, wie erwähnt, inkrementell neue Objekte zu *konstruieren* und dabei bereits definierten Kode wiederzuverwenden. Im Gegensatz dazu drückt die Untertypbeziehung Eigenschaften der externen *Verwendung* von Objekten oder Termen im allgemeinen, nicht deren Implementierung, aus. Daß es sich also um verschiedene Begriffe handelt, die nicht verwechselt gehören, ist allgemein anerkannt, allerdings nicht so allgemein, daß nicht Sprachen wie beispielsweise C^{++} oder Eiffel Typen und Klassen, und damit Untertypen und Vererbung, gleichsetzten. Zur Diskussion von Untertypen vs. Vererbung siehe auch neben vielen weiteren [Sny86] [BHJ$^+$87] [CHC90] [LP91].

Objekte als Records In einer oft verwendeten Analogie betrachtet man Objekte als den Record ihrer Methoden [CW85], wobei der Methodenaufruf der Recordselektion entspricht. In erster Annäherung paßt die Intuition der Untertypbeziehung für Objekte auch gut mit den Untertypregeln für Records zusammen: zwei Recordtypen stehen in Untertyprelation, wenn der kleinere alle Felder des größeren, und möglicherweise mehr, enthält und darüberhinaus die Typen der gemeinsamen Felder in kovarianter Untertyprelation stehen (die Felder l_i des Records sind unterschiedlich und ihre Reihenfolge spielt keine Rolle):

$$\frac{S_i \;\leq\; T_i \qquad \text{für alle } 1 \leq i \leq n}{\{l_1{:}S_1, \ldots, l_n{:}S_n, l_{n+1}{:}S_{n+1}, \ldots, l_{n+m}{:}S_{n+m}\} \;\leq\; \{l_1{:}T_1, \ldots, l_n{:}T_n\}}$$

So gut wie alle typisierten Kalküle für objektorientierte Sprachen verwenden diese Analogie. Um das Zusammenspiel von Untertypisierung, Vererbung mit dynamischer Methodenbindung und Überschreiben von Methoden, sowie Kapselung geeignet modellieren zu können, muß diese vereinfachende Gleichsetzung von Objekten und Records allerdings verfeinert werden.

Speziell in objektbasierten Sprachen, bei denen die Records oder Objekte das direkt, über die Recordselektion, das heißt, über den Methodenaufruf, hinaus manipuliert werden, endet die Analogie zischen Records und Objekten. Recorderweiterung, bei der einem bestehenden Record ein neues Feld hinzugefügt wird, erfordert, eine Hälfte der oberen Untertypregel für Records aufzugeben, den Teil nämlich, der festlegt, daß die Records des Untertyps mehr Felder als die Records des Obertyps besitzen dürfen. (Dies wird als Untertypisierung der Breite für Records bezeichnet.) Behielte man die volle Untertypregel bei, würde einem Subsumption gestatten, einen Objekt eine Methode hinzuzufügen, die es bereits besitzt, was zu einem Laufzeitfehler führte. Um mit dem Hinzufügen von Methoden oder Recorderweiterungen umgehen zu können, wurden eine Anzahl von sogenannten Record-Kalkülen vorgeschlagen. Der Hauptgedanke in Zusammenhang mit Recorderweiterungen ist dabei, in das Typsystem Information über die *Ab*wesenheit von Methoden in einem Objekt aufzunehmen, um sicherzustellen, daß diese Methoden gefahrlos zur Laufzeit dem Objekt hinzugefügt werden können.

Um andererseits das Überschreiben von Methoden modellieren zu können, also die Ersetzung einer Methode durch eine andere gleichen Namens, erzwingt die Aufgabe der zweiten Hälfte der oberen Regel, daß nämlich die Typen der gemeinsamen Felder beider Records in Untertypbeziehung stehen — dies wird als Untertypbeziehung der Tiefe für Records bezeichnet. Anstelle dessen muß man, erlaubt man das Überschreiben von Methoden zu Laufzeit, auf der Identität der Typen der gemeinsamen Methoden bestehen.

Für die angesprochenen Probleme wurden eine Reihe von Record-Kalkülen insbesondere in Zusammenhang mit Typinferenz vorgeschlagen und untersucht, die unter anderem das Überschreiben von Recordfeldern, die Erweiterung von Records oder die Konkatenation von Records unterstützen [Wan87a] [Wan88] [Wan87b] [Car88a] [Wan91] [CM91] [Rém89] [Rém92] [HP91] [Car92] [Wan94] [Aba94] [JM93] [Hen94] [GJ96] [Zwa96] ... Die Selbstreferenz dadurch, daß eine Methode weiteren Methoden — einschließlich sich selbst — desselben Objektes aufrufen kann, kann mit rekursiven Recordtypen modelliert werden. Beispielsweise werden Klassen in [Coo87] [Coo89] durch Funktionen, sogenannte Objektgeneratoren, und Objekte als rekursiv definierte Records als Fixpunkt des Generators dargestellt. Weitere Arbeiten zu rekursiven Records sind neben anderen [CP89] [Red88] [KR94] [Bru92] [CHC90] [CCHM89] [Mit90b]. Ein komplexer Sprachentwurf mit Objekte als rekursiven Records findet sich in [Bru94].

Überladen Anstelle als einem bestimmten Objekt zugehörig, kann man eine Methode auch als eigenständige Einheit betrachten (und implementieren), wobei der Kode, der bei Aufruf der Methode ausgeführt wird, von den Objekten abhängt, an denen sie aufgerufen wird. Indem die Methoden eine Existenz außerhalb der Objekte besitzen, kommt aus Sicht der Methode keinem der Objekte eine Sonderstellung zu und der selektierte Kode kann von mehr als einem Objekt abhängen. Dies ist als *Mehrfachdispatch*, zum Beispiel in CLOS [KG89] [DG87], bekannt. Der Methodenaufruf wird als als Funktion und die Objekte als seine Parameter betrachtet, das heißt, der Methodenaufruf ist eine überladene Funktion. Eine Erweiterung des einfach typisierten λ-Kalküls um Untertypen und überladene Funktionen (genannt $\lambda\&$) wird in [Cas93] [CGL95] präsentiert, um Mehrfachdispatch oder Multimethoden zu modellieren. Das System $F_{\leq}^{\&}$ [Cas95] stellt die entsprechende Erweiterung auf den Fall zweiter Ordnung dar. Eine Metasprache λ_{object}, die auf $\lambda\&$ basiert, wird in [Cas93] untersucht. Vergleiche auch Castagna [Cas97] für eine umfangreiche Behandlung objektorientierte Sprachen mit Multimethoden.

Matching Matching, geschrieben als $S <\# T$, ist eine Relation auf Typen. Sie ähnelt der Untertyprelation, ist aber allgemeiner.[4] Insbesondere Subsumption, der Kern der Untertypbeziehung, ist keine Eigenschaft des Matchings. Die Relation wurde von Bruce und anderen vorgeschlagen, und erlaubt die Definition von Klassen, für deren Unterklassen man die größere Freiheit besitzt, nicht nur aus allen Untertypen zu wählen sondern aus allen matchenden Typen. Dies ist allgemeiner, denn es erlaubt die Konstruktion von Unterklassen, deren Instanzen nicht notwendigerweises Typen tragen, die Untertypen der des Typs der Oberklasse sind [Bru94] [Bru96b] [BFP97] [AC96a]. Die Form der beschränkte Polymorphie bei der der Parameter nicht durch einen Obertyp als obere Grenze sondern durch einen matchenden Typ beschränkt ist, nennt man *match-beschränkte* Polymorphie oder beschränktes Matching. Matching läßt sich im durch Untertypen und Polymorphie höherer Ordnung repräsentieren [AC96a]. Beispiele für Sprachenentwürfe mit Matching sind Toil [BSvG93] und polyTOIL [BSvG95], statisch typisierte Sprachen mit Untertypen und Matching sowie imperativen Anteilen. Im einem neueren Vorschlag Loom [BFP97] [Bru96b] werden Untertypen vollständig zu Gunsten von Matching aufgegen. Weitere Sprachen mit match-beschränkter Polymorphie sind Theta [LCD$^+$94] [DGLM95] und School [RIR93].

F-beschränkte Polymorphie F-beschränkte Polymorphie wurde in [CCHM89] [CHC90] als eine Sonderform der Untertypisierung höherer Ordnung vorgeschlagen. Die obere Schranke S in einem polymorphen Typ $\forall A {\leq} S.T$ kann dabei die Typvariable A frei enthalten, wodurch sich das S wie eine Funktion von Typen nach Typen, also wie ein Typoperator, verhält. Eine polymorphe Funktion diesen Typs erwartet als Typargument ein U mit $U \leq [U/A]T$. Vergleiche hierzu auch [BCGŠ91] [KLMM94]. Sprachen mit F-beschränkter Polymorphie sind unter anderem Sather [OL92] und Pizza.

Objektkalküle

Die angesprochenen funktionalen Kalküle behandeln Objekte als einen abgeleiteten Begriff. Alternativ wurden eine Reihe von Formalismen mit primitiven Objekten als vorgeschlagen. Die eindrucksvollste Familie derartiger Kalküle — untypisierte Kalküle, Kalküle erster und höherer Ordnung — findet sich in Abadis und Cardellis Buch [AC96b]. Von einem untypisierten Objektkalkül ς-calculus [AC94] ausgehend, der Methodenüberschreibung und dynamische Methodenbindung,[5] aber keine Metho-

[4]Die Matching-Relation darf nicht mit dem aus einigen höheren Programmiersprachen wie beispielsweise ML bekannte *pattern matching* verwechselt werden.

[5]Das Symbol ς des ς-calculus ersetzt λ als Variablenbinder um daran zu erinnern, das dynamische Methodenbindung sich von der statischen Bindung von Funktionen unterscheidet.

denerweiterung umfaßt, wird der objektbasierte Kalkül um Typsysteme erster und höherer Ordnung ausgebaut. Eine weitere stärkere Variante dieser Familie findet sich in [Liq97].

Andere Kalküle formalisieren nebenläufige, objektorientierte Programmierung. Ein Beispiel dafür ist die Sprache Pict [PT97b][PT97c], deren Fundament der π-Kalkül [MPW92], eine Prozeßalgebra für mobile Kommunikation, ist. Statische Typisierung ist auch für Prozeßkalküle eine wichtige Fragestellung [PS96]. Während die Typisierungsdisziplin in einem sequentiellen, funktionalen Rahmen dafür sorgt, daß Funkionen nicht auf unpassende Argumente angewendet werden, interagieren Prozesse in komplexerer Weise mit ihrer Umwelt, nämlich mittels Kommunikation und Synchronisation. Dabei werden statische Typsystem verwendet, um sicherzustellen, daß Kommunikationskanäle nur gemäß ihrer Deklaration verwendet werden. In Pict beispielsweise ist eine mächtige Version von $F^\omega_\leq$ implementiert, die rekursive Typen, partielle Typinferenz, und *pattern matching* unterstützt. Weitere nebenläufige objektorientierte Kalküle sind [Nie92] [DF96] [HT91] [HT92] [VT93].

1.3 $F^\omega_\leq$: Untertypisierung höherer Ordnung

In den letzten zehn Jahren gab es eine beträchtliche Anzahl von Untersuchungen und Vorschlägen für typisierte, funktionale Kalküle, d.h. typisierte λ-Kalküle, die flexibel und ausdrucksstark genug sind, um als mathematische Grundlage für objektorientierter Sprachen zu dienen. Die vorliegende Arbeit beschäftigt sich mit einem der wichtigsten Vertreter dieser Kalküle, bekannt als $F^\omega_\leq$ ("F-omega-sub"). Dieser typisierte λ-Kalkül höherer Ordnung mit Untertypen läßt sich am einfachsten als Kombination seiner Unterkalküle verstehen: der reine polymorphe λ-Kalküle — das System F—, den Kalkül F^ω und den polymorphen λ-Kalkül mit Untertypen $F_\leq$.

Als Ausgangspunkt dient *Système F* oder der polymorphe λ-Kalkül von Girard und Reynold [Gir71] [Gir72] [Rey74]. Dieses System erweitert den einfach typisierten λ-Kalkül von Church [Chu40] um parametrische Polymorphie, indem es Typabstraktion $\lambda A.t$ und Typapplikation $t\,T$ auf Termebene erlaubt. (Dabei ist A eine Typvariable und T ein Typ.) Fügt man weiter *Typoperatoren*, also Funktionen von Typen nach Typen, hinzu, erhält man F^ω [Gir72], ein System mit Polymorphie höherer Ordnung. Erweitert man andererseits den polymorphen λ-Kalkül um Untertypen und beschränkte Quantifikation, erhält man $F_\leq$; Cardelli und Wegners Sprache „Bounded Fun“ [CW85] ist eine bekannte Variante dieses Systems. Die metatheoretischen Eigenschaften von $F_\leq$ sind in der Literatur vielfach untersucht worden, beispielsweise in [CMMŠ94] [Ghe91] [CG92] [BCGŠ91] [Pie94].

Die Kombination schließlich von Untertypen und Typoperatoren ergibt das System $F^\omega_\leq$ [Car90] [Mit90b], den Kalkül mit Untertypisierung höherer Ordnung. Er ist ausdrucksstark genug, um Klassenverwebung mit dynamischer Methodenbindung, Kapselung von Objekten und Untertyppolymorphie zu modellieren. Er wurde als unterliegender Basiskalkül für klassenbasierte objektorientierte Sprachen in der Tradition von Smalltalk vorgeschlagen [PT95b] [PT94] [HP95b]. Objekte werden in diesem Kalkül als Elemente existentiellen Typs dargestellt, wodurch ihr interner Zustand und die Implementierung ihrer Methoden nach außen unsichtbar bleibt [MP88]. Typoperatoren machen es möglich, nicht alleine Objekte, sondern auch Objektschnittstellen oder Signaturen als Funktionen von Typen nach Typen eigenständig zu repräsentieren, nämlich als eine Funktion, die den Repräsentierungstyp eines Objektes als Argument nimmt und deren Schnittstelle über dieser Repräsentierung zurückliefert.

Fragen der Implementierung und von Erweiterungen von $F_{\leq}$, wie beispielsweise partielle Typinferenz für Typen zweiter Ordnung, de Bruijn Indizes und ein Mechanismus zur Syntaxerweiterung werden in [Car93] diskutiert. [Cra97] behandelt entsprechend den Fall höherer Ordnung. Der KML-Compiler [Cra96b] beruht auf dem explizit typisierten Kalkül λ^K, der Records, rekursive Typen und Potenzarten umfaßt. Das Modell von Pierce und Turner [PT94] kann in FCP [Jon97] repräsentiert werden, einem Kalkül mit Polymorphie erster Ordnung, Typinferenz und abstrakten Datentypen.

Für eine ausführlichere Darstellung des Objektmodells von $F_{\leq}^{\omega}$ siehe [PT94] und [HP95b]. Ein Vergleich verschiedener Objektmodelle und Kodierungren findet sich in den Übersichtsartikeln [FM96] und [BCP96]. In [PT93] wird eine Verfeinerung des Objektmodells von $F_{\leq}^{\omega}$ untersucht, welches „freundliche“ Funktionen mittels partiell-abstrakter Typen (siehe [MP88]) darstellt. Erweiterungen um rekursive Typen finden sich in [AC93] [MPS86] [CC91].

Für semantische, denotationelle *Modelle* von Untertypkalkülen mit Polymorphie wurden partielle Äquivalenzrelationen (PERs) verwendet. Ein wichtiges Papier über die denotationelle Semantik von Untertypen mittels partiellen Äquivalenzrelationen ist Bruce und Longo [BL90]. Ein PER-Modell für $F_{\leq}^{\omega}$ findet sich in [BM92] [AP90], ähnlich ein Modell für $F_{\leq}$ mit positiver Untertyprelation in [HP95a]. Ein Modell, welches auf Umwandlungsfunktionen beruht, wird in [BCGŠ91] ausgearbeitet. Weiteres Material über Modelltheorie in diesem Zusammenhang findet sich in [Pho90] [Ama91] [MV96] [FMRS92].

Varianten

Die Kalküle $F_{\leq}$, $F_{\leq}^{\omega}$ und ähnliche treten in verschiedenen Varianten auf. Ein wichtiges Unterscheidungsmerkmal ist dabei die Version der Untertypregel für universelle Quantifikation. Eine einfache Version der All-Regel wurde in [CW85] für die Sprache “Bounded Fun” oder “Kernel Fun” vorgeschlagen. Die Regel fordert, daß die Oberschranken der verglichenen Alltypen übereinzustimmen haben:

$$\frac{\Gamma, A{\le}U \;\vdash\; S_2 \;\le\; T_2}{\Gamma \;\vdash\; \mathit{All}(A{\le}U)S_2 \;\le\; \mathit{All}(A{\le}U)T_2} \qquad (\text{S-ALL-KERNEL})$$

Eine stärkere, semantisch gleichfalls sinnvolle Formulierung, hier mit S-ALL-FULL bezeichnet, gestattet den kontravarianten Vergleich beider Obergrenzen:

$$\frac{\Gamma, A{\le}T_1 \;\vdash\; S_2 \;\le\; T_2 \qquad \Gamma \;\vdash\; T_1 \;\le\; S_1}{\Gamma \;\vdash\; \mathit{All}(A{\le}S_1)S_2 \;\le\; \mathit{All}(A{\le}T_1)T_2} \qquad (\text{S-ALL-FULL})$$

Ghelli [Ghe95] zeigte, daß ein zuvor entwickelter Untertyp-„algorithmus“ in Anwesenheit der vollen Untertypregel divergieren kann; Pierce [Pie94] bewies zusätzlich, daß diese Regel die Untertyprelation für $F_{\le}$ unentscheidbar macht. Weitere Untersuchungen der Implikationen dieser Regel finden sich in Ghelli [Ghe93]. In Curien und Ghelli [CG92] wird die Metatheorie der vollen Version von $F_{\le}$ mit Hilfe von Umwandlungsfunktionen, sogenannten *coercions*, untersucht, die als eine explizite Repräsentierung der Ableitungen für Untertypurteile angesehen werden können. Mittels Techniken der Beweisnormalisierung erhalten sie einen korrekten und vollständigen Semialgorithmus für $F_{\le}$'s Untertyprelation. Eine ähnliche Untersuchung in [BCGŠ91] bildet $F_{\le}$ in den polymorphen λ-Kalkül F mit Recordtypen ab.

Die schwächere S-ALL-KERNEL-Regel führt zu Systemen mit wesentlich angenehmeren beweistheoretischen Eigenschaften als Systeme mit der komplexeren All-Regel. Die einfachere Variante wurde beispielsweise in [Com95b] und [PS97] für Polymorphie höherer Ordnung verwendet. Ein weiterer Vergleich beider Regeln und in Anwesenheit beschränkt-existentieller Quantifikation findet sich in [GP97].

Die vorliegende Arbeit schlägt eine weitere Variante der Untertypregel für Alltypen vor. Einerseits ist es unser Ziel, soweit es geht die positiven, beweistheoretischen Eigenschaften der Kernvariante von $F^{\omega}_{\le}$ beizubehalten. Auf der anderen Seiten wird sich herausstellen, daß durch die Erweiterung um Polarisierung die Forderung nach Identität der Obergrenzen aufgegeben und durch die schwächere Bedingung einer gegenseitigen Untertypbeziehung ersetzt werden muß:

$$\frac{\Gamma \;\vdash\; S_1 \;\gtrless\; T_1 \;\in\; K_1 \quad \Gamma, A{\le}S_1{:}K_1 \;\vdash\; S_2 \;\le\; T_2 \;\in\; K}{\Gamma \;\vdash\; \mathit{All}(A{\le}S_1{:}K_1)S_2 \;\le\; \mathit{All}(A{\le}T_1{:}K_1)T_2 \;\in\; \star} \qquad (\text{S-ALL-POL})$$

In Zusammenhang mit polarisierter Untertypisierung ist dies eine echt stärkere Regel, denn anders als für punktweise Untertypisierung von Applikationen, impliziert die gegenseitige Untertypbedingung nicht β-Gleichheit.

Erweiterungen

In Hofmann und Pierce [HP95a] wird eine „positive“ Variante von Kern-$F_{\le}$ mit dem Ziel vorgeschlagen, in funktionalem Rahmen die Semantik des Überschreibens von

Werten zu beschreiben. Üblicherweise kann die Untertyprelation $S \leq T$ als das Vorhandensein einer Extraktionsfunktion des Typs $S \rightarrow T$ interpretiert werden, die Elemente des kleineren Typs in solche des größeren umwandelt. Hofmann und Pierce weichen von dieser „orthodoxen“ Sichtweise ab indem sie für ein Untertypbeziehung $S \leq T$ zwischen S und T nicht alleine die Existenz der Standardextraktionsfunktion, sondern zusätzlich eine Überschreibungsfunktion des Typs $S \rightarrow T \rightarrow S$ fordern. Zu diesem Zweck führen sie für jedes Paar von Typen eine Konstante $put[S,T]$ auf Termebene ein. Die zusätzlichen Forderungen führen zu einer restriktiveren Interpretation der Untertyprelation als gewöhnlich; beispielsweise gibt es in dieser stärkeren Interpretation keine nichttriviale Untertypbeziehung zwischen existentiellen Typen, also Objekttypen. Darüberhinaus ist die Untertypregel für Pfeiltypen eingeschränkt, in dem die Argumenttypen auf beiden Seiten übereinzustimmen haben. Unter diesen Einschränkungen erweitert und vereinfacht das Objektmodell das entsprechende aus [PT94], indem sich beispielsweise die Darstellung der Vererbung wesentlich vereinfacht, da die Überschreibfunktionen bereits Bestandteil der Untertyprelation sind und nicht mehr explizit vom Programmierer angegeben werden müssen. Die stärkere Interpretation erlaubt es auch, die Verifikation objektorientierte Programme entlang ihrer Struktur zu organisieren, indem sich Eigenschaften der Objekte von den Oberklassen auf die Unterklassen übertragen lassen. Dieses Modell der positiven Untertypisierung wurde in [HNSS98] in dem typentheoretischen Beweisprüfer Lego kodiert. Unter Ausnutzung der Korrespondenz zwischen Aussagen und Typen, bekannt als Curry-Howard-Isomorphie [CF58] [How80] [GLT89], wird eine Beweiskomponente in die Klassendefinition mitaufgenommen, wodurch Objekte mit Beweismethoden ausgerüstet werden, die in exakter Analogie zu den funktionalen Methoden durch an der Methodenschnittstelle gekapselt sind; nach außen sichtbar sind nur die Eigenschaften des Objekts, ihre Beweise bleiben nach außen verborgen.

Schnittypen Eine Reihe von Untersuchungen verallgemeinern $F_{\leq}$ oder $F^{\omega}_{\leq}$ um *Schnittypen.* Die entsprechenden Kalküle werden als $F_{\wedge}$ (“F-meet”) [Pie97b] [Ma92] im Fall zweiter Ordnung und als $F^{\omega}_{\wedge}$ (“F-omega-meet”) [Com95a] in Anwesenheit von Typoperatoren bezeichnet. Begrifflich ist mit dem Schnitt von S und T (geschrieben als $S \wedge T$ für den binären Schnitt von S und T) die Menge aller Terme, die beide Typen tragen. Verwendet man Schnittypen zur Beschränkung polymorph-parametrischer Funktionen, so läßt sich damit ausdrücken, daß eine Unterklasse von mehr als einer Oberklasse erbt. Diese Modell von *Mehrfachvererbung* durch Schnittypen wird in [CP96] entwickelt. Das System $F^{\omega}_{\wedge}$ (“F-omega-meet”) ist eine echte Verallgemeinerung von $F^{\omega}_{\leq}$, bei der der maximale Typ *Top* als der leere Schnitt verstanden werden kann. Für die Behandlung von Schnittypen zusammen mit Typoperatoren verweise ich auf die These von Compagnoni [Com95b]. Ein Sprachentwurf, der auf Schnittypen beruht, ist Reynolds Forsythe [Rey96].

Abhängige Typen $F^{\omega}_{\leq}$ ist eine imprädikative Typentheorie, bei der Terme sowohl von Termen als auch von Typen abhängen und bei der Typen als Parameter Typen akzeptieren (in diesem Fall werden sie als Typoperatoren bezeichnet.) Systeme, in denen Typen von Termen abhängen können, werden als Systeme mit *abhängigen* Typen bezeichnet. Dies erlaubt die Bildung von Typen die uniform für eine Menge von Termen definiert sind, beispielsweise den Typ aller Arrays der Länge n, wobei n eine natürliche Zahl sei. Die Stärke dieser Systeme reicht weit über das Arraybeispiel hinaus. Sie erlauben es, logische Eigenschaften zu formulieren und bilden die Grundlage vieler typentheoretischer Beweisprüfer wie Lego [leg97], Coq [CCF$^+$95], Isabelle [Pau93], Nuprl [C$^+$86] et. al. Für eine systematische Klassifikation der verschiedenen λ-Kalküle nach Art und Kombination ihrer Abhängigkeiten zwischen Termen und Typen im sogenannten λ-Kubus sei auf Barendregts Handbuchbeitrag [Bar92] verwiesen.

Die Kombination von abhängigen Typen und Untertypisierung hat in jüngster Zeit steigende Aufmerksamkeit auf sich gezogen. Aspinall und Compagnoni [AC96c] untersuchen $\lambda P_{\leq}$, einen Kalkül erster Ordnung mit abhängigen Typen (bekannt als λP oder $\lambda\Pi$) und erweitert um Untertypen, und weisen seine Entscheidbarkeit nach. Chen [Che97a] behandelt eine unterschiedliche Formulierung des selben Systems, welches er als $\lambda\Pi_{\leq}$bezeichnet. Die Kombination von abhängigen Typen und Untertypen führt zu einer zyklischen Abhängigkeit des Typsystems, des Untertypsystems und des Artsystem, sodaß die beweistheoretischen Probleme denen ähneln, die bei der Untersuchung des polarisierten $F^{\omega}_{\leq}$ auftreten. Für eine genauere Diskussion der Unterschiede verweise ich auf den Schluß der Arbeit.

1.4 Weiterführende Literatur

Eine gewisse Vertrautheit mit typisierten, funktionalen und objektorientierten Kalkülen wird bei der Lektüre der Arbeit hilfreich sein. Ich kann folgende Bücher oder Artikel von breiterer Perspektive empfehlen.

Umfangreiches Material über typisierte funktionale Sprachen, das heißt, typisierte λ-Kalküle, ihre operationellen und denotationellen Semantiken, findet sich in den Büchern von Mitchell [Mit96] und Gunter [Gun92]. Obwohl beide Bücher ihren Schwerpunkt auf funktionale Sprachen legen, enthalten sich auch Abschnitte über Untertyppolymorphie. Insbesondere Mitchell diskutiert das funktionale Recordmodell objektorientierter Sprachen sowie beschränkte Polymorphie, einschließlich F-beschränkter Polymorphie und Polymorphie höherer Ordnung. Mit ähnlicher Themenstellung befaßt sich Thompsons Buch [Tho91], doch mit größerem Gewicht auf Typentheorie als eigenständige logische Disziplin und weniger in Hinblick auf Sprachentwurf. Thompson diskutiert demzufolge auch Typsysteme mit abhängigen

Typen, wie sie in Beweisprüfern Anwendung finden. Zwei lesenswerte Handbuchartikel über Typsysteme für Programmiersprachen sind Mitchell [Mit90a] und, erst jüngst erschienen, Cardelli [Car97]. Eine breite Diskussion über den Nutzen von Typen in Programmiersprachen mit vielen Verweisen auf bestehende Programmiersprachen findet sich in Cardelli [Car91]. Das Papier diskutiert auch eine Reihe von Implementierungsfragen.

Während es eine Anzahl von Standardreferenzen zu den Grundlagen funktionaler Sprachen gibt, sind einführende theoretische Werke über objektorientierte Sprachen rarer. Eine umfassende, wichtige Abhandlung in diesem Zusammenhang ist Abadi und Cardellis Buch [AC96b], in dem die Autoren ihren Standpunkt, der grundlegende Begriff zur Untersuchung objektorientierter Sprachen seien Objekte, nicht Funktionen, anhand einer Reihe von untypisierten und typisierten Objektkalkülen unterschiedlicher Ausdrucksmächigkeit darstellen. Die Kalküle unterstützen das Überschreiben von Methoden, Untertypen und dynamische Methodenbindung; komplexere Merkmale wie Klassen und Vererbung werden durch Kodierung in die Basiskalküle erklärt. Castagnas soeben erschienenes Buch [Cas97] unternimmt eine ähnliche Darstellung für objektorientierte Sprachen mit Multimethoden. Die mathematischen Formalismen sind hierbei λ-Kalküle mit Überladen. Typsysteme für klassenbasierte objektorientierte Sprachen mit Schwerpunkt auf Typinferenz werden in Palsberg und Schwarzbachs Buch [PS94] untersucht.

Verschiedene Ansätze zur Behandlung der kontroversen Frage binärer Methoden werden in [BCC+96] von einer Reihe von Forschern auf diesem Gebiet diskutiert. Auf der Basis von $F^{\omega}_{\leq}$ mit rekursiven Typen werden in [BCP96] verschiedene bekannte Objektkodierungen verglichen. Im Übersichtspapier [Bru96b], stellt Bruce eine Reihe von typisierten Kalkülen für objektorientierte Sprachen einander gegenüber und argumentiert zu Gunsten von match-beschränkter Polymorphie. Er illustriert seinen Standpunkt mit einer Vielzahl von Beispielen und Verweisen auf existierende objektorientierte Sprachen und Sprachentwürfe. Viele einflußreiche Originalveröffentlichungen über funktionale Kalküle für objektorientierte Sprachen sind im Band [GM94] zusammengestellt und neu veröffentlicht.

1.5 Der Beitrag der Arbeit

Der Kalkül $F^{\omega}_{\leq}$, ein polymorpher λ-Kalkül mit Untertypen und Typoperatoren, wurde als Basiskalkül für typisierte, objektorientierte Sprachen vorgeschlagen. Die in der Literatur untersuchten Versionen unterstützen in der Regel nur punktweise Untertypisierung von Typoperatoren, das heißt, zwei Applikationen $S\ U$ und $T\ U$ stehen in Untertypbeziehung, wenn S und T dies tun. Cardelli präsentiert in seiner häufig zitierten, unveröffentlichten Note $F^{\omega}_{\leq}$ in einer allgemeineren Form. Sie geht über die punktweise Untertypisierung von Applikationen hinaus, indem sie zwischen monotonen und antimonotonen Operatoren unterscheidet. Beispielsweise ist die Applikation $T\ U_1$ ein Untertyp von $T\ U_2$, falls $U_1 \leq U_2$ und T monotoner Operator ist.

Meine These erweitert $F^{\omega}_{\leq}$ um polarisierte Applikation, entwickelt seine Metatheorie, und weist die Entscheidbarkeit des polarisierten $F^{\omega}_{\leq}$ nach. Die Einführung polarisierter Applikationen führt zu wechselseitiger Abhängigkeit des Untertyp– und des Artsystems. Dies steht im Gegensatz zum unpolarisierten Fall, bei dem Untertypableitungen von Artableitungen abhängen, aber nicht umgekehrt. Um die Entscheidbarkeit des System nicht zu verlieren, stellt die Arbeit eine Verallgemeinerung der bekannten Untertypregel für All-Typen vor, die auf der Identität der Oberschranken der verglichenen Alltypen beharrt. Die Verallgemeinerung im polarisierten Fall besteht darin, daß dies zu einer Forderung nach gegenseitiger Untertypbeziehung der Oberschranken abgeschwächt wird. Soweit mir bekannt, ist die Arbeit die erste beweistheoretische Abhandlung einer polarisierten Erweiterung von $F^{\omega}_{\leq}$.

1.6 Gliederung

Teil I enthält das Material über $F^{\omega}_{\leq}$ ohne Polarisierung. Der Hauptteil der Arbeit findet sich in Teil II, in dem der Untertypkalkül höherer Ordnung um Polaritätsinformation erweitert und seine Beweistheorie untersucht wird. Der Aufbau beider Teile ist streckenweise parallel und gelegentlich sind nur die Beweise für den komplizierteren polarisierten Kalkül aufgeführt. Das abschließende Kapitel in Teil III diskutiert verwandte Arbeiten und schlägt Richtungen für weitere Forschungen vor. Die meisten Beweise wurden in Anhang in Teil V aufgenommen.

1.7 Veröffentlichungen

Das Material über $F^{\omega}_{\leq}$ in Teil I wurde bereits zusammen mit Benjamin Pierce in dem Technischen Bericht [SP94], in einer Konferenzversion und einer Journalversion [PS97] veröffentlicht worden. Dieser Teil wurde mit geringeren Änderungen in Aufbau und Darstellung, sowie einigen Vereinfachungen mit in die Arbeit aufgenommen. Das Material wurde aus Teil II wurde auf dem jährlichen „Types“-Treffen „Subtyping, Inheritance, and Modular Development of Proofs“ in Durham, 1997, und auf dem Kolloquium über die „Grundlagen der Programmierung“ in Avendorf, Fehmarn, vorgestellt.

Inhaltsverzeichnis

Curriculum Vitæ

Persönliche Daten

geboren:	6. Juni 1965 in Bad Honnef
Familienstand:	ledig

Schulausbildung

Grundschule:	August 1971 – Juli 1975 Volksschulen in Lemgo, Suttrop und Freising
Gymnasium:	September 1975 – Juli 1977 Joseph-Hofmiller-Gymnasium in Freising September 1977 – Juni 1984 Rhön-Gymnasium Bad Neustadt a.d. Saale
Abschluß:	Abitur

Universitätsausbildung

Studium:	WS 86/87 – WS 90/91 Friedrich-Alexander-Universität Erlangen-Nürnberg Hauptfach Informatik, Nebenfach Physik
Vordiplom:	November 1988
Ferienakademie:	Teilnahme an der Ferienakademie der TU München und der FAU Erlangen-Nürnberg im September 1990 Thema: “Anwendungen verteilter Programmierung”
Hauptdiplom:	Oktober 1991
Diplomarbeit:	November 1991 – März 1992 „Ein vollständiges Beweissystem für Hennessy-Milner-Logik mit Rekursion“
Abschluß:	Ende März 1992
Dissertation:	„Polarized Higher-Order Subtyping“

Sonstiges

Zivildienst:	August 1984 – März 1986 im Kreiskrankenhaus Mellrichstadt
Sprachkenntnisse:	Englisch, Französisch, Spanisch, Latinum

Part V

Proofs

Appendix A

Reduction Relations

This chapter first proves some basic facts about $\beta\top$-reduction. In the second more complex section we prove strong normalization of a general reduction relation, combining $\beta\top$-reduction and replacement of type variables by their upper bound in a given context (called Γ-reduction) wich will be needed for the termination of the subtyping algorithm. It closely resembles the corresponding section in the paper [PS97]. The argument here also benefited much from discussions with Adriana Compagnoni. The Γ-reduction relation, wich we will define below in Definition A.10, had to be adapted in order to work for the polarized case, as well.

A.1 Properties of $\beta\top$-reduction and substitution

This section contains some easy facts about the $\beta\top$-reduction relation we need a various places throughout this work, for both the pure and the polarized case.

Definition A.1 (Parallel reduction) *Parallel* reduction is the smallest relation closed under the following rules:

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

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

$$\frac{S \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} S' \qquad T \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} T'}{S \to T \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} S' \to T'} \qquad \frac{S \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} S' \qquad T \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} T'}{\mathit{All}(A{\le}S{:}K)\ T \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} \mathit{All}(A{\le}S'{:}K)\ T'}$$

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

Fact A.2

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

Lemma A.3

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

In the proof of Lemma A.3 we will make use of the following property of substitution.

Fact A.4 If $A \neq A'$ and $A' \notin fv(S)$, then $[S/A]([T/A']U) = [[S/A]T/A']([S/A]U)$.

A useful consequence of the third part of Lemma A.3 is that in a reduction sequence a reduction step reducing the outermost redex can be performed at the beginning of the sequence without changing the result.

Corollary A.5 (Outermost reduction) Assume $(Fun(A{:}K)S)T \longrightarrow^{*}_{\beta\top} U$ with $S \longrightarrow^{*}_{\beta\top} S'$ and $T \longrightarrow^{*}_{\beta\top} T'$. If $U \neq (Fun(A{:}K)S')T'$ then $[T/A]S \longrightarrow^{*}_{\beta\top} U$.

Lemma A.6 (Diamond property for $\twoheadrightarrow_{\beta\top}$) For all types S, S_1, and S_2 with $S \twoheadrightarrow_{\beta\top} S_1$ and $S \twoheadrightarrow_{\beta\top} S_2$, there exists a type S_3, such that $S_1 \twoheadrightarrow_{\beta\top} S_3$ and $S_2 \twoheadrightarrow_{\beta\top} S_3$.

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

Proof: By a simple extension of the standard argument. (see Barendregt [Bar84]).

□

Corollary A.7 (Church-Rosser for $\longrightarrow^*_{\beta\top}$) For all types S, S_1 and S_2 with $S \longrightarrow^*_{\beta\top} S_1$ and $S \longrightarrow^*_{\beta\top} S_2$ there is a type S_3 with $S_1 \longrightarrow^*_{\beta\top} S_3$ and $S_2 \longrightarrow^*_{\beta\top} S_3$.

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

The following property will be used in the proof of Lemma 5.62.

Lemma A.8 If $S \twoheadrightarrow_{\beta\top} S_1$ and $S \longrightarrow^*_{\beta\top} S_2$, then there is a type S_3 such that $S_1 \longrightarrow^*_{\beta\top} S_3$ and $S_2 \twoheadrightarrow_{\beta\top} S_3$.

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

Proof of Lemma A.8: By induction on the length of the sequence $S \longrightarrow^*_{\beta\top} S_2$, using Fact A.2 and the Church-Rosser property of parallel reduction.

Base step: $n = 0$
Trivial.

Induction step: $n > 0$ and $S \longrightarrow_{\beta\top} S'_2 \longrightarrow^{n-1}_{\beta\top} S_2$
With $\longrightarrow_{\beta\top} \subset \twoheadrightarrow_{\beta\top}$ we get by the Church-Rosser property for $\twoheadrightarrow_{\beta\top}$ that there exists a type S'_3 with $S_1 \twoheadrightarrow_{\beta\top} S'_3$ and $S'_2 \twoheadrightarrow_{\beta\top} S'_3$. The result follows from the induction hypothesis. □

Finally we state normalization of $\beta\top$-reduction.

Lemma A.9 (Strong $\beta\top$-normalization) Suppose $\Gamma \vdash S \in K$. Then there is no infinite $\beta\top$-reduction from S.

Proof: The result can be proven translating the kinds of $F_{\leq}^{\omega}$ into kinds of F^{ω}. The translation F from $F_{\leq}^{\omega}$- to F^{ω}-type is given by $F(\mathit{Top}(\star)) = \mathit{All}(A)A$, $F(\mathit{Top}(K_1 \to K_2)) = \mathit{Fun}(A{:}K_1)(F(\mathit{Top}(K_2)))$; types of other forms are homomorphically translated into F^{ω}. Since *Top*-types of arrow kind are translated to type operators in F^{ω}, each $\beta\top$-step of a type of $F_{\leq}^{\omega}$ can be mimicked by a β-step in F^{ω}, and the assumption of an infinite $\beta\top$-reduction sequence in $F_{\leq}^{\omega}$ would contradict strong termination of β-reduction in F^{ω} [Gir72]. □

A.2 Γ-reduction

Next we generalize the promotion relation $\uparrow_\Gamma$ (cf. Definition 3.1 on page 30) so that a type variable occuring inside may be replaced by its respective upper bound, either declared in the context or given by the variable's binding occurrence in the type itself. The definition of Γ-reduction is very similar to the one of [PS97]. The only rule which is different is the one for type application. Here, we are more restrictive in that applications may only be Γ-reduced by promoting their *head* variable. Besides simplifying some of the proofs a bit, this restriction is not important as far as pure $F_{\leq}^{\omega}$ is concerned. However, it will allow us to use the relation in the polarized setting, as well.

Definition A.10 Single-step Γ-*reduction* is the least family of relations closed under:

$$\frac{A\ T_1 \dots T_n \ \uparrow_\Gamma\ \Gamma(A)\ T_1 \dots T_n}{A\ T_1 \dots T_n \longrightarrow_\Gamma \Gamma(A)\ T_1 \dots T_n} \qquad \frac{S \longrightarrow_{(\Gamma,\, A:K)} S'}{\mathit{Fun}(A{:}K)S \longrightarrow_\Gamma \mathit{Fun}(A{:}K)S'}$$

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

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

Single-step $\beta\top\Gamma$-reduction ($\longrightarrow_{\beta\top\Gamma}$) is defined as the union $\longrightarrow_{\beta\top} \cup \longrightarrow_\Gamma$. The corresponding multi-step reductions are defined as usual.

In the polarized setting we cannot allow promotion of arbitrary type variables to their upper bound (as would be possible for the pure case, cf [PS97]) since for termination of polarized subtyping we will need not simply strong termination of $\beta\top\Gamma$-reduction, but strong normalization of the combined relation $\longrightarrow_{\beta\top\Gamma} \equiv$. Equivalence $\equiv$ acts like identity (up-to renaming of bound variables) on types, except that it ignores syntactic substructures of the type occurring constantly. This means that the effect of a Γ-step for a type variable occurring constantly can be reversed by $\equiv$, rendering already the relation $\longrightarrow_{\Gamma} \equiv$ cyclic. The above definition only promotes variables occurring in head position of an application but not inside, which is an easy way to ensure that a variable occuring constantly never gets promoted, because the only position for a variable syntactically present in a type but occurring constantly is in the argument of a constant operator. Alternatively, one might have defined a Γ-reduction relation which takes polarity information into account, but Definition A.10 is simpler.

Lemma A.11 (Strong Γ-normalization) If $\Gamma \vdash S \in K$, then there is no infinite Γ-reduction from S.

Proof: Observe that well-kinded types and contexts and also well-scoped: there is no way for a definition to refer to itself recursively. So we can assign a weight to each well-scoped type that is the sum of weights of its variables, where the weight of the variable is one greater than the weight of its upper bound (either in the context or in the binding occurance in the type). Each Γ-step reduces this measure. $\square$

Lemma A.12 (Subject reduction) If $\Gamma \vdash S \in K$ and $S \longrightarrow^{*}_{\beta\top\Gamma} S'$, then $\Gamma \vdash S' \in K$.

Proof of Lemma A.12: By induction on the length $S \longrightarrow^{*}_{\beta\top\Gamma} S'$, with an inner induction on the definition of single-step $\beta\top\Gamma$-reduction, using preservation of kinding under $\beta\top$-reduction and under promotion (Lemma 3.14 and 3.18). $\square$

Lemma A.13

1. If $(T_1 \to T_2) \longrightarrow^{*}_{\Gamma} V$, then $V = (V_1 \to V_2)$ where $T_1 \longrightarrow^{*}_{\Gamma} V_1$ and $T_2 \longrightarrow^{*}_{\Gamma} V_2$.
2. If $\mathit{All}(A{\le}T_1)T_2 \longrightarrow^{*}_{\Gamma} V$, then $V = \mathit{All}(A{\le}V_1)V_2$ where $T_2 \longrightarrow^{*}_{(\Gamma,\, A\le T_1)} V_2$ and $T_1 \longrightarrow^{*}_{\Gamma} V_1$.
3. If $\mathit{Fun}(A{:}K)T_2 \longrightarrow^{*}_{\Gamma} V$, then $V = \mathit{Fun}(A{:}K)V_2$ where $T_2 \longrightarrow^{*}_{(\Gamma,\, A:K)} V_2$.
4. If $T_1\ T_2 \longrightarrow^{*}_{\Gamma} V$, then $V = V_1\ T_2$ where $T_1 \longrightarrow^{*}_{\Gamma} V_1$.

Proof: We give the proof in detail for part (2); the rest is similar, but simpler. The form of $V = \mathit{All}(A{\le}V_1)V_2$ is immediate by the definition of $\longrightarrow_{\Gamma}$. For the rest of part (2), we prove the more refined statement

> if $T_1 \longrightarrow^{*}_{\Gamma} U_1$ and $T_2 \longrightarrow^{*}_{(\Gamma,\, A\le T_1)} U_2$ and $\mathit{All}(A{\le}U_1)U_2 \longrightarrow^{*}_{\Gamma} \mathit{All}(A{\le}V_1)V_2$,
> then $T_1 \longrightarrow^{*}_{\Gamma} V_1$ and $T_2 \longrightarrow^{*}_{(\Gamma,\, A\le T_1)} V_2$,

by induction on the length of reduction from $\mathit{All}(A{\le}U_1)U_2$ to $\mathit{All}(A{\le}V_1)V_2$.

Case: $All(A{\leq}U_1)U_2 = All(A{\leq}V_1)V_2$
Immediate.

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

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

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

Lemma A.14 (Weak diamond property for Γ-reduction)

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

Proof: By induction on the form of T.

Case: $T = A\ T_1 \ldots T_n$
Then $V = U = \Gamma(A)\ T_1 \ldots T_n$ and we may take $W = \Gamma(A)\ T_1 \ldots T_n$.

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

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

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

Set $W_2 = T_2$ and we are done.
Subcase: $U = All(A{\leq}T_1)U_2$ and $V = All(A{\leq}T_1)V_2$
We must find a W_2 such that

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

The existence of such a W_2 is given by the induction hypothesis.
Subcase: $U = All(A{\leq}T_1)U_2$ and $V = All(A \leq V_1)T_2$
Set $W_1 = V_1$. Then we must show:

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

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

Other cases:
Straightforward. □

Lemma A.15 (Church-Rosser for Γ-reduction) If $\Gamma \vdash T \in K$, then

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

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

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

Lemma A.16 If $T_1 \longrightarrow^*_{\beta\top} U_1$ and $T_2 \longrightarrow^*_{(\Gamma,\, A \leq T_1)} V_2$, then there is some W_2 such that:

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

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

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

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

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

If $V_2' \longrightarrow_{(\Gamma,\, A\leq T_1)} V_2$ by contracting a redex other than A, then $V_2' = V_2'[B]$ and $V_2 = V_2'[\Gamma(B)]$. In reducing from V_2' to W_2', the variable B can neither vanish nor be copied, as we do not Γ-reduce type variables other than in head position of an application. Hence $W_2' = W_2'[B]$ and we can choose W_2 as $W_2'[\Gamma(B)]$.

Similarly, if $V_2' = V_2'[A]$ with $V_2 = V_2'[T_1]$ and $W_2' = W_2'[A]$, take $W_2 = W_2'[U_1]$. The sequence $V_2 = V_2'[T_1] \longrightarrow^*_{\beta\top} W_2'[U_1]$ follows by the assumption $T_1 \longrightarrow^*_{\beta\top} U_2$, and $W_2'[A] \longrightarrow_{(\Gamma,\, A\leq U_1)} W_2'[U_1]$ by definition of Γ-reduction. Since $\beta\top$-reduction cannot affect type variables occuring in head position of an application, the last Γ-step is still possible. □

The next lemma establishes a confluence property for Γ- and $\beta\top$-reductions. The proof is similar to that of Lemma A.14. This lemma and Lemma A.15 jointly handle the crucial step in the strong normalization argument that follows.

Lemma A.17 ($\beta\top$-reduction and Γ-reduction)

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

Proof of Lemma A.17: By induction on the definition of $T \longrightarrow_{\beta\top} U$.

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

The type T is in Γ-normal-form and, choosing $W = U$, the case is immediate.

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

Again, T is in Γ-normal-form, and the case is analogue to the previous one.

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

By Lemma A.13, type V has the form $All(A{\le}V_1)V_2$, with $T_1 \longrightarrow^*_{\Gamma} V_1$ and with $T_2 \longrightarrow^*_{(\Gamma,\, A\le T_1)} V_2$. Apply the induction hypothesis to find a W_1 with

$$\begin{tikzcd}
T_1 \arrow[r, "\beta\top"'] \arrow[d, "\Gamma"', "*"] & U_1 \arrow[d, "*"', "\Gamma"] \\
V_1 \arrow[r, "+", "\beta\top"'] & W_1.
\end{tikzcd}$$

By Lemma A.16, there is some W_2 such that:

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

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

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

By Lemma A.13, type V has the form $All(A{\le}V_1)V_2$, with $T_1 \longrightarrow_{\Gamma} V_1$ and with $T_2 \longrightarrow_{(\Gamma,\, A\le T_1)} V_2$. We must show

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

which follows directly from the induction hypothesis.

Other cases:
Similarly, using parts (2) to (4) of Lemma A.13. □

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

Lemma A.18 ($\beta\top$-postponement) Assume $\Gamma \vdash T \in K$. If $T \longrightarrow_{\beta\top} U \longrightarrow_{\Gamma} X \longrightarrow^{\infty}_{\beta\top\Gamma} \ldots$, then there is some V_0 such that $T \longrightarrow_{\Gamma} V_0 \longrightarrow^{\infty}_{\beta\top\Gamma} \ldots$.

For the proof, we need a simple fact:

Fact A.19 If $S \longrightarrow_{\beta\top} T \longrightarrow_{\Gamma} U$, then $S \longrightarrow_{\Gamma} U'$ for some U'. (That is, the redex that is contracted between T and U is a residual of a redex already present in S.)

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

Proof of Lemma A.18: By Fact A.19, there is some V_0 such that:

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

By Lemma A.17, there is some V_1 such that

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

Since $U \longrightarrow_{\Gamma} X$, we can now use Church-Rosser for Γ-reduction (Lemma A.15):

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

We can continue in this way, applying either A.15 or A.17 to successive elements of the infinite reduction beginning from X to obtain an infinite sequence of multi-step $\beta\top\Gamma$-reductions on the right:

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

But the sequence of reductions on the left must contain infinitely many $\beta\top$-steps (otherwise it would have an infinite Γ-tail), so Lemma A.17 also tells us that infinitely many of the individual multi-step reductions on the right are nonempty. The reduction $T \longrightarrow_\Gamma V_0 \longrightarrow^*_{\beta\top\Gamma} V_1 \cdots \longrightarrow^*_{\beta\top\Gamma} *$ is the desired one. $\square$

Proposition A.20 (Strong $\beta\top\Gamma$-normalization) If the type S is well-kinded in Γ, then there is no infinite $\beta\top\Gamma$-reduction from S.

Proof: Assume, for a contradiction, that $\mathcal{R}$ is an infinite $\beta\top\Gamma$-reduction beginning from S. Let $\mathcal{R}_0 = \mathcal{R}$. Now repeat the following process as long as possible to construct a sequence $\mathcal{R}_1$, $\mathcal{R}_2$, ... of infinite $\beta\top$-reductions, all starting from S:

> If $\mathcal{R}_i$ contains no Γ-reduction that is immediately preceded by a $\beta\top$-reduction, then stop. Otherwise, form $\mathcal{R}_{i+1}$ from $\mathcal{R}_i$ by using Lemma A.18 repeatedly to move the first such Γ-reduction before any $\beta\top$-reduction.

Note that all of the $\mathcal{R}_i$ are infinite and that the first i steps in each $\mathcal{R}_i$ are all Γ-reductions. Now, there are two possibilities:

- The sequence of $\mathcal{R}$'s eventually terminates, having reached some $\mathcal{R}_n$ in which all Γ-reductions precede the first $\beta\top$-reduction. But this means that $\mathcal{R}_n$ contains only Γ-reductions, contradicting Lemma A.11, or has an infinite tail consisting only of $\beta\top$-reductions, contradicting Lemma A.9.

- The sequence of $\mathcal{R}$'s is infinite. But since each $\mathcal{R}_i$ begins with at least i Γ-reductions, we can use this to exhibit an infinite Γ-reduction beginning from S, contradicting Lemma A.11. □

Appendix B

Proofs for $F^{\omega}_{\leq}$

This chapter contains the proof for the pure calculus of $F^{\omega}_{\leq}$ We elide proofs which are straightforward.

B.1 Kinding

Proof of decidability of kinding (Lemma 3.10 on page 32): By straightforward induction we can prove that the two kinding systems and the two definitions of context well-formedness are equivalent. In each direction, we only have to consider the rule for variables, since all other rules coincide.

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

By Lemma 3.4(2) $\vdash \Gamma_1, A{\leq}T, \Gamma_2$ *ok* as subderivation. So by the induction hypothesis, $\vdash_{\mathcal{A}} \Gamma_1, A{\leq}T, \Gamma_2$ *ok* and $\Gamma_1, A{\leq}T, \Gamma_2 \vdash_{\mathcal{A}} T \in K$. Repeated application of Lemma 3.9 yields $\Gamma_1 \vdash_{\mathcal{A}} T \in K$.

Case K-Tvar-A: $\Gamma_1 \vdash_{\mathcal{A}} T \in K$ and $\Gamma \vdash_{\mathcal{A}} \Gamma_1, A{\leq}T, \Gamma_2$ *ok*

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

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

Proof of Lemma 3.11 on page 32: By straightforward induction. □

Proof of Lemma 3.13 on page 32: Both parts are proved simultaneously by induction on derivations. □

Proof of Lemma 3.14 on page 33: We prove the lemma not for arbitrary sequences of reduction steps but for one parallel step. Proceed by induction on the length of derivation in both parts of the lemma. □

Proof of Corollary 3.15 on page 33: By the Church-Rosser property (A.7 on page 179), uniqueness of kinding (3.6 on page 31), and subject reduction for kinds (3.14 on page 33). □

B.2 The reducing system

Proof of Lemma 3.18 on page 34: By straightforward induction on the length of derivation. □

Proof of Lemma 3.19 on page 34: By inspection of the rules of the reducing system (Definition 3.16) and uniqueness of kinding. □

Proof of Lemma 3.20 on page 34: By induction over the length of derivations, using preservation of kinding under reduction and under promotion (Lemma 3.14 and Lemma 3.18), and the generation Lemma 3.5 for kinds at various places. □

Proof of Lemma 3.21 on page 34: By induction on the length of derivation, using well-kindedness of subderivations. In the cut-free system, only the rules for reflexivity, for maximal types, or for application are applicable. Reflexivity and the *Top*-rule are immediate, the application rule by induction. □

B.3 Subject reduction for subtyping

Proof of Lemma 3.22 on page 35: By induction on the length of derivation of $\Gamma \vdash S \leq T$. □

Proof of Lemma 3.23 on page 35: Part 1 by induction on the derivation of $\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K)T$.

Case R-Refl:

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

We know that

$$\begin{array}{lll} S\ U \longrightarrow^*_{\beta\top} (\mathit{Fun}(A{:}K)T')\ U \longrightarrow_{\beta} [U/A]T' & \text{and} \\ T\ U \longrightarrow^*_{\beta\top} (\mathit{Fun}(A{:}K)T')\ U \longrightarrow_{\beta} [U/A]T'. \end{array}$$

By preservation of kinding under reduction and generation for kinds $\Gamma \vdash U \in K$. By well-kindedness of subderivations and the generation lemma for kinds further $\Gamma, A{:}K \vdash T' \in K_2$. Since by Lemma A.3 $[U/A]T \longrightarrow^*_{\beta\top} [U/A]T'$, the case follows with the expansion lemma.

Case R-Promote:

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

By definition of promotion, $S \longrightarrow^*_{\beta\top} A'\ V_1 \ldots V_n \uparrow_{\Gamma} \Gamma(A')\ V_1 \ldots V_n = S'$. By preservation of kinding under reduction and promotion, the induction hypothesis applies, yielding $\Gamma \vdash_{\mathcal{C}} \Gamma(A')\ V_1 \ldots V_n\ U \leq [U/A]T'$, so the result follows with R-Promote.

Case R-Abs:

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

The remaining cases are solved similarly.

In the second part of the lemma, R-Refl and R-Abs are solved analogously to the corresponding cases in the first part.

Case R-Top:

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

By preservation of kinding under reduction, uniqueness of kinding, and twice the generation lemma for kinds $K' = K \to K_2$. Since also $T\ U$ is well-kinded, we get $T\ U \longrightarrow^*_{\beta\top} \mathit{Top}(K \to K_2)\ U \longrightarrow_{\top} \mathit{Top}(K_2)$. So the case follows from $\Gamma \vdash (\mathit{Fun}(A{:}K)S)\ U \in K_2$ using R-Top. □

Proof of Lemma 3.24 on page 35: By induction on the depth of inference, similar to the proof of 3.23. □

Proof of Lemma 3.25 on page 35: The first part for the reduction step on the right hand side by induction over the derivation of $\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K)T)\ U$.

Most cases are straightforward. With the exception of application rule, we get by Corollary A.5 that $(\mathit{Fun}(A{:}K)T)\ U \longrightarrow^*_{\beta\top} W$ implies $[U/A]T \longrightarrow^*_{\beta\top} W$. Hence in order to derive $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T$, we can directly use the derivation of the original statement $\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K)T)\ U$.

The case for application is more difficult, since by contracting the outermost redex, the use of the application rule, that might have led in a last derivation step to the statement $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T$, can be rendered impossible.

Case R-App:
$$\frac{S \longrightarrow^{*}_{\beta\top} S_1\ V \qquad (\mathit{Fun}(A{:}K)T)\ U \longrightarrow^{*}_{\beta\top} T_1\ V \qquad \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1}{\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K)T)\ U}$$

We distinguish, whether the outermost redex $(\mathit{Fun}(A{:}K)T)\ U$ gets contracted in the reduction sequence or not. If it is, the case is easy and similar to the ones for the other rules:

Subcase: $(\mathit{Fun}(A{:}K)T)\ U \longrightarrow^{*}_{\beta\top} (\mathit{Fun}(A{:}K)T')\ U' \longrightarrow_{\beta} [U'/A]T' \longrightarrow^{*}_{\beta\top} T_1\ V$
By Lemma A.2 also $[U/A]T \longrightarrow^{*}_{\beta\top} [U'/A]T' \longrightarrow^{*}_{\beta\top} T_1\ V$, the case is immediate by the application rule and preservation of kinding under reduction.

Subcase: $(\mathit{Fun}(A{:}K)T)\ U \longrightarrow^{*}_{\beta\top} (\mathit{Fun}(A{:}K)T'_1)\ V$
where $\mathit{Fun}(A{:}K)T \longrightarrow^{*}_{\beta\top} \mathit{Fun}(A{:}K)T'_1 = T_1$ and $U \longrightarrow^{*}_{\beta\top} V$. So we are given $\Gamma \vdash_{\mathcal{C}} S_1 \leq \mathit{Fun}(A{:}K)T'_1$. By well-kindedness of subderivations and part of 1 Lemma 3.23 $\Gamma \vdash_{\mathcal{C}} S_1\ V \leq [V/A]T'_1$. Since $[U/A]T \longrightarrow^{*}_{\beta\top} [V/A]T'_1$ (Lemma A.2), the case follows with the expansion lemma.

The second part of the lemma, dealing with an outer reduction step on the left-hand side, is shown by induction on the derivation of $\Gamma \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K)S)\ U \leq T$.

The cases for R-Refl, R-Arrow, R-All, and R-Abs are symmetric to the corresponding cases in the first part of the Lemma; R-Top and R-Promote are solved by simple induction. The case for application is solved analogously to the respective ones in the first part, using part 2 of Lemma 3.23. □

Proof of Lemma 3.26 on page 35: Both parts of the lemma by induction on the length of derivation. The properties of the parallel reduction relation allows (Lemma A.5) to use induction in all cases, especially in the case of R-Promote.

Case R-Refl:
$$\frac{S \longrightarrow^{*}_{\beta\top} U \qquad T \longrightarrow^{*}_{\beta\top} U}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma A.6 there exists a type U' such that $U \twoheadrightarrow_{\beta\top} U'$, $S' \longrightarrow^{*}_{\beta\top} U'$, and $T \longrightarrow^{*}_{\beta\top} U'$, so the case follows by reflexivity.

Case R-Top:
$$\frac{T \longrightarrow^{*}_{\beta\top} \mathit{Top}(K) \qquad \Gamma \vdash S \in K}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

The case is immediate by R-Top and by subject reduction for kinding.

Case R-Promote:

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

By definition of promotion $W = A\ S_1 \dots S_n \uparrow_\Gamma \Gamma(A)\ S_1 \dots S_n = U$. By Lemma A.6 there are types $S'_1, \dots, S'_n$ with $S_i \twoheadrightarrow_{\beta\top} S'_i$ such that:

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

The case follows by induction, the fact that $\Gamma(A)\ S_1 \dots S_n \twoheadrightarrow_{\beta\top} \Gamma'(A)\ S'_1 \dots S'_n$, and R-Promote.

Case R-All:

$$\frac{S \longrightarrow^*_{\beta\top} All(A{\leq}U)S_2 \qquad T \longrightarrow^*_{\beta\top} All(A{\leq}U)T_2 \qquad \Gamma,\ A{\leq}U \vdash_{\mathcal{C}} S_2 \leq T_2}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma A.6 and the definition of parallel reduction there are two types U' and S'_2 such that:

$$\begin{tikzcd}
S \arrow[r, two heads, "\beta\top"'] \arrow[d, "*"', "\beta\top"] & S' \arrow[d, "*"', "\beta\top"] \\
All(A{\leq}U)S_2 \arrow[r, two heads, "\beta\top"'] & All(A{\leq}U')S'_2.
\end{tikzcd}$$

By induction $\Gamma',\ A{\leq}U' \vdash_{\mathcal{C}} S'_2 \leq T_2$. As $T \longrightarrow^*_{\beta\top} All(A{\leq}U)T_2 \twoheadrightarrow_{\beta\top} All(A{\leq}U')T_2$, the case follows with Lemma A.2 and R-All.

Case R-Abs, R-Arrow:

Similar.

Case R-App:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ U \qquad T \longrightarrow^*_{\beta\top} T_1\ U \qquad \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1}{\Gamma \vdash_{\mathcal{C}} S \leq T}$$

By Lemma A.6 there exists a type R such that:

$$\begin{tikzcd}
S \arrow[r, twoheadrightarrow, "\beta\top"'] \arrow[d, "*"', "\beta\top"] & S' \arrow[d, "*"', "\beta\top"] \\
S_1\ U \arrow[r, twoheadrightarrow, "\beta\top"'] & R
\end{tikzcd}$$

By well-kindedness of subderivations, S_1 and T_1 are well-kinded in Γ. We have to distinguish according to the form of S_1; the interesting case is, where S_1 is a type operator.

Subcase: $S_1 = A$ or $S_1 = V_1\ V_2$
By definition of parallel reduction, $R = S_1'\ U'$ with $S_1 \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} S_1'$ and $U \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} U'$. Thus the case follows by induction, R-App, using $T \longrightarrow^{*}_{\beta\top} T_1\ U'$ (Lemma A.2).

Subcase: $S_1 = \mathit{Top}(K)$
By well-kindedness of subderivations and generation for kinds $K = K_1 \to K_2$. By Lemma A.6 there exists a type R such that:

$$\begin{tikzcd}
S \arrow[r, twoheadrightarrow, "\beta\top"'] \arrow[d, "*"', "\beta\top"] & S' \arrow[d, "*"', "\beta\top"] \\
\mathit{Top}(K_1 \to K_2)\ U \arrow[r, twoheadrightarrow, "\beta\top"'] & R
\end{tikzcd}$$

We distinguish whether the parallel step from $\mathit{Top}(K_1 \to K_2)\ U$ to R contracts the $\top$-redex or not. If $\mathit{Top}(K)\ S_2 \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} \mathit{Top}(K_1 \to K_2)\ S'$, the case follows by induction. If $\mathit{Top}(K_1 \to K_2)\ S_2 \longrightarrow_{\top} \mathit{Top}(K_2)$, the case follows by Lemma 3.24.

Subcase: $S_1 = \mathit{Fun}(A{:}K)S_1'$
We are given $(\mathit{Fun}(A{:}K)S_1)\ S_2 \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} R$. By definition of parallel reduction we can distinguish the following two subcases: one, where the outer redex does not get contracted in the parallel step $\mathit{Fun}(A{:}K)S_1'\ S_2 \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} R$, the second one, where the outer redex disappears.

Subsubcase: $R = (\mathit{Fun}(A{:}K)S_1'')\ U'$
$S_1 = \mathit{Fun}(A{:}K)S_1' \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} \mathit{Fun}(A{:}K)S_1''$
$U \longrightarrow\!\!\!\!\!\twoheadrightarrow_{\beta\top} U'$
This easier subcase is solved by induction and R-App, using subject reduction for kinds and the generation lemma.

Subsubcase: $R = [U'/A]S''_1$
$S'_1 \twoheadrightarrow_{\beta\top} S''_1$
$U \twoheadrightarrow_{\beta\top} U'$
We are given $\Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S'_1 \leq T_1$. By induction $\Gamma' \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S''_1 \leq T_1$ and hence with R-App:

$$\frac{\Gamma' \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K)S''_1 \leq T_1}{\Gamma' \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K)S''_1)\ U' \leq T_1\ U'}$$

By Lemma 3.25, we can perform one outer reduction step on the left-hand side, obtaining $\Gamma' \vdash_{\mathcal{C}} [U'/A]S''_1 \leq T_1\ U$. Since $S' \longrightarrow^*_{\beta\top} (\mathit{Fun}(A{:}K)S''_1)\ U' \longrightarrow_{\beta} [U'/A]S''_1$ and $T \longrightarrow^*_{\beta\top} T_1\ U'$, we conclude with the expansion lemma $\Gamma' \vdash_{\mathcal{C}} S' \leq T$.

The second part of the lemma, dealing with a parallel reduction step on the right-hand side, is analogous. The critical cases are again handled by Lemma 3.25. The case for R-Promote is easier. $\square$

Proof of Corollary 3.27 on page 36: If we can do one $\twoheadrightarrow_{\beta\top}$-step, we can do many. By Fact A.2, $\twoheadrightarrow^*_{\beta\top}$ equals $\longrightarrow^*_{\beta\top}$. $\square$

Proof of Corollary 3.28 on page 36: By Corollary 3.27 and Lemma 3.19. $\square$

B.4 Cut elimination

Proof of Lemma 3.31 on page 38: By induction on the length of the reduction relation $\nearrow^*_{\Gamma} = (\longrightarrow^!_{\beta\top} \uparrow_{\Gamma})^n$.

For $n = 0$ we have $S = A\ S_1 \ldots S_n$, since S must be in normal form, and the case is immediate. Since for a type U with $U \longrightarrow_{\beta\top} U'$ or $U \uparrow_{\Gamma} U'$ we know $U\ T \longrightarrow_{\beta\top} U'\ T$ respectively $U\ T \uparrow_{\Gamma} U'\ T$, the induction step can be proven by an inner induction on the number of $\beta\top$-steps resp. the $\uparrow_{\Gamma}$-step. $\square$

Proof of Lemma 3.32 on page 38: By definition, $\nearrow^*_{\Gamma}$ consists of a sequence of $\longrightarrow_{\beta\top}$– and $\uparrow_{\Gamma}$–steps and the result follows by a sequence of instances of the promotion rule, well-kindedness of subderivations and the expansion lemma. $\square$

Proof of Lemma 3.33 on page 39: By induction on the derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T$; with strong derivations and in absence of transitivity, only the rules of reflexivity, of promotion, or of application are available. $\square$

B.5 A subtyping algorithm for $F^{\omega}_{\leq}$

Before we prove soundness and completeness, first some auxiliary lemmas.

Lemma B.1 Assume the types S, S', T, and T' well-kinded in Γ. If $\Gamma \vdash_{\mathcal{O}} S \leq T$, then $\Gamma \vdash_{\mathcal{O}} S' \leq T'$

Proof: By induction on the length of derivation, using well-kindedness of subderivations (Lemma 3.20), S-Conv, and S-Trans. □

Lemma B.2 Assume S well-kinded in Γ. If $S \uparrow_{\Gamma} S'$, then $\Gamma \vdash_{\mathcal{O}} S \leq S'$.

Proof: By definition of promotion, S must be of the form $A\ S_1 \ldots S_n$. Proceed by induction on the length of the kinding derivation $\Gamma \vdash A\ S_1 \ldots S_n \in K$. The case for K-TVar is immediate with S-TVar. The one for K-Arrow-E follows by induction and S-App. □

Lemma B.3 Assume S and T well-kinded in Γ. Then $\Gamma \vdash_{\mathcal{CS}} S \leq T$, iff. $\Gamma \vdash_{\mathcal{A}} S^{!} \leq T^{!}$.

Proof: The "only-if"-direction is obvious. So assume $\Gamma \vdash_{\mathcal{CS}} S \leq T$ and proceed by induction on the length of derivation. The only interesting case is the one for application.

Case R-App:

$$\frac{S \longrightarrow^{!}_{\beta\top} S_1\ U \qquad T \longrightarrow^{!}_{\beta\top} T_1\ U \qquad \Gamma \vdash_{\mathcal{CS}} S_1 \leq T_1}{\Gamma \vdash_{\mathcal{CS}} S \leq T}$$

$T_1\ U$ must be of the form $A\ U_1 \ldots U_n\ U$ for some $n \geq 0$. By Lemma 3.33 $S \nearrow^{*}_{\Gamma} \longrightarrow^{!}_{\beta\top} A\ U_1 \ldots U_n\ U$. This means, $\Gamma \vdash_{\mathcal{A}} S^{!} \leq A\ U_1 \ldots U_n\ U$ follows by an appropriate number of instances of A-Promote, preceded by A-Refl. □

Proof of soundness and completeness (Proposition 3.36): As the system of strong, cut-free derivations and the subtyping algorithm and the coincide by the previous lemma, it suffices to show soundness and completeness of the $\vdash_{\mathcal{CS}}$-system. Soundness is proved by straightforward induction on the length of derivation, using well-kindedness of subderivations, Lemma B.1, and in the case of promotion, also Lemma B.2.

For completeness, proceed once again by induction on the depth of inference.

Case S-Conv:

$$\frac{S =_{\beta\top} T \qquad \Gamma \vdash S, T \in K}{\Gamma \vdash_{\mathcal{O}} S \leq T}$$

By Church-Rosser and strong normalization of $\beta\top$-reduction, we have $S \longrightarrow^{!}_{\beta\top} U$ and $T \longrightarrow^{!}_{\beta\top} U$ for some type U, and the result follows by R-Refl.

Case S-Trans:
$$\frac{\Gamma \vdash_{\mathcal{O}} S \leq U \qquad \Gamma \vdash_{\mathcal{O}} U \leq T \qquad \Gamma \vdash_{\mathcal{O}} U \in K}{\Gamma \vdash_{\mathcal{O}} S \leq T}$$

By induction $\Gamma \vdash_{\mathcal{CS}} S^! \leq U^!$, and $\Gamma \vdash_{\mathcal{CS}} U^! \leq T^!$. Thus by cut-elimination (Proposition 3.34) $\Gamma \vdash_{\mathcal{CS}} S^! \leq T^!$.

Case S-Top:
$$\frac{\Gamma \vdash S \in K}{\Gamma \vdash_{\mathcal{O}} S \leq \mathit{Top}(K)}$$

By preservation of kinding under reduction an R-Top.

Case S-TVar:
$$\frac{}{\Gamma \vdash_{\mathcal{O}} A \leq \Gamma(A)}$$

By R-Promote and R-Refl.

Case S-App:
$$\frac{\Gamma \vdash_{\mathcal{O}} S \leq T}{\Gamma \vdash_{\mathcal{O}} S\ U \leq T\ U}$$

By well-kindedness of subderivations and induction $\Gamma \vdash_{\mathcal{CS}} S^! \leq T^!$. Using the application rule R-App we get a cut-free, but not necessarily strong derivation for $\Gamma \vdash_{\mathcal{C}} S^!\ U \leq T^!\ U$. Finally by Lemma 3.29 $\Gamma \vdash_{\mathcal{CS}} (S\ U)^! \leq (T\ U)^!$.

The remaining cases by simple induction. □

Appendix C

Proofs for Polarized $F^{\omega}_{\leq}$

C.1 Kinding

Proof of Lemma 5.1 on page 65: By rule K-Refl, the subkind-relation is reflexive. The relation $\leq$ on polarities is a partial order by definition. That transitivity and antisymmetry are inherited by the subkinding relation can be proven by induction over the length of derivation. □

Proof of Lemma 5.2 on page 66: By straightforward induction. □

Proof of Lemma 5.4 on page 66: The first part by inspection of the context rules, the second and third part by straightforward induction. □

Proof of Lemma 5.5 on page 67: In the first part we are given $\Gamma \vdash T\ ?_A$, assuming a derivation in the system of Definition 5.3. We assume further $? \leq ?'$ and show by induction that also $\Gamma \vdash T\ ?'_A$ by a derivation in the new system. The cases where $? = ?'$ are trivial, so assume in the following $? < ?'$.

The rules for variables are immediate, also the rule dealing with the free occurrence of type variables. The remaining rules are solved by induction, using monotonicity of $\neg$ and $\times$ on the domain of polarities.

In the second part of the lemma we are given $\Gamma \vdash T \in K$ in the system of Definition 5.3 on page 66 and a kind K' with $K \leq K'$.

Case K-TVar-Sub:

$$\frac{\mathit{kind}_\Gamma A \leq K \qquad \vdash \Gamma\ \mathit{ok}}{\Gamma \vdash A \in K}$$

By transitivity of $\leq$ on kinds (Lemma 5.1).

Case K-Top-Sub:

$$\frac{K'_1 \leq K_1 \qquad \Gamma \vdash \mathit{Top}(K_2) \in K'_2}{\Gamma \vdash \mathit{Top}(K_1 \to^{?} K_2) \in K'_1 \to^{?'} K'_2}$$

From the assumption $K'_1 \to^{?'} K'_2 \leq K'$ we get $K' = K''_1 \to^{?''} K''_2$ with $K''_1 \leq K'_1$ and $K'_2 \leq K''_2$. By induction thus $\Gamma \vdash \mathit{Top}(K_2) \in K''_2$. Further by transitivity of $\leq$ on kinds $K''_1 \leq K_1$, so the case follows by K-Top-Sub.

The cases for arrow types and universally quantified types are trivial since they posses $\star$ as unique kind.

Case K-Arrow-I-Sub:

$$\frac{\begin{array}{c} K_1 \leq K''_1 \\ \Gamma, A{:}K''_1 \vdash T\ ?_A \qquad \Gamma, A{:}K''_1 \vdash T \in K_2 \end{array}}{\Gamma \vdash \mathit{Fun}(A{:}K''_1)T \in K_1 \to^{?} K_2}$$

and we are given $K_1 \to^{?} K_2 \leq K'$. By definition of subkinding, we get $K' = K'_1 \to^{?'} K'_2$, with $K'_1 \leq K_1$ and $K_2 \leq K'_2$ and $? \leq ?'$. By transitivity of subkinding (Lemma 5.1) $K'_1 \leq K''_1$. By induction $\Gamma, A{:}K''_1 \vdash T\ ?'_A$ and $\Gamma, A{:}K''_1 \vdash T \in K'_2$, and we conclude by K-Arrow-I-Sub:

$$\frac{\begin{array}{c} K'_1 \leq K''_1 \\ \Gamma, A{:}K''_1 \vdash T\ ?'_A \qquad \Gamma, A{:}K''_1 \vdash T \in K'_2 \end{array}}{\Gamma \vdash \mathit{Fun}(A{:}K''_1)T \in K'_1 \to^{?'} K'_2}$$

Case K-Arrow-E:

$$\frac{\Gamma \vdash S \in K_1 \to^{?} K_2 \qquad \Gamma \vdash T \in K_1}{\Gamma \vdash S\ T \in K_2}$$

Assume $K_2 \leq K'_2$. Thus $K_1 \to^{?} K_2 \leq K_1 \to^{?} K'_2$, by induction $\Gamma \vdash S \in K_1 \to^{?} K'_2$, and we can finish by K-Arrow-E. □

Proof of Lemma 5.7 on page 67: We have to two directions to show, both by induction over the length of derivations.

We start with the proof of correctness of the rules of Definition 5.3, i.e., the new presentation does not derive more statements than the original system. The rules of the form $\vdash \Gamma\ ok$ and $\Gamma \vdash S\ ?_A$ are solved inspection or by straightforward induction. For the statements of the form $\Gamma \vdash S \in K$ we can use K-Subsumption to prove the correctness. We only show the rule for arrow introduction:

Case K-Arrow-I-Sub:

$$\frac{\begin{array}{c} K_1 \leq K_1' \\ \Gamma, A{:}K_1' \vdash T\ ?_A \qquad \Gamma, A{:}K_1' \vdash T \in K_2 \end{array}}{\Gamma \vdash \mathit{Fun}(A{:}K_1')T \in K_1 \rightarrow^? K_2}$$

By induction $\Gamma,\ A{:}K_1' \vdash T\ ?_A$ and $\Gamma,\ A{:}K_1' \vdash T \in K_2$ in the original system. By K-Arrow-I therefore $\Gamma \vdash \mathit{Fun}(A{:}K_1')T \in K_1' \rightarrow^? K_2$. Since $K_1' \rightarrow^? K_2 \leq K_1 \rightarrow^? K_2$, we get a derivation for $\Gamma \vdash \mathit{Fun}(A{:}K_1')T \in K_1 \rightarrow^? K_2$ by K-Subsumption.

For completeness we once again proceed by induction on the depth of inference, using Lemma 5.5. The subsumption rule for the occurrence of type variables is solved by induction and part 1 of Lemma 5.5. The case for $\Gamma \vdash A +_A$ is immediate by the corresponding rule of Definition 5.3; likewise the one for $\Gamma \vdash T \circ_A$ where A does not occur freely in T. The rules for arrow-types, universally quantified types, and type application by straightforward induction.

For statements of the form $\Gamma \vdash T \in K$ most cases are solved by induction. For instance K-Arrow-I is covered by induction, reflexivity of $\leq$ on kinds, and K-Arrow-I-Sub. K-Top-Sub is analogous.

Case K-Subsumption:

$$\frac{K' \leq K \qquad \Gamma \vdash T \in K'}{\Gamma \vdash T \in K}$$

By induction we get $\Gamma \vdash T \in K'$ in the new system and by part 2 of Lemma 5.5 on page 67 also $\Gamma \vdash T \in K$. □

Proof of the generation Lemma 5.8 on page 67: By inspection of the rules of the kinding system from Definition 5.3. □

Proof of Lemma 5.10 on page 68: By induction on the definition of subkinding. □

Proof of Lemma 5.11 on page 68: By induction on the combined length of derivation for $\Gamma \vdash T \in K_1$ and $\Gamma \vdash T \in K_2$, using Lemma 5.10. □

Proof of Lemma 5.12 on page 68: Assume two kinds K' and K'' with $K = \mathit{erase}(K') = \mathit{erase}(K'')$. Proceed by induction on the syntactic structure of kind K.

Case: $K = \star$
Thus also K' and K'' equal $\star$ and the result is trivial.

Case: $K = K_1 \to K_2$
K' is of the form $K'_1 \to^{?'} K'_2$ with $\mathit{erase}(K'_1) = K_1$ and $\mathit{erase}(K'_2) = K_2$. For the second kind likewise $K'' = K''_1 \to^{?''} K''_2$ where $\mathit{erase}(K''_1) = K_1$ and $\mathit{erase}(K''_2) = K_2$. By induction, we get two greatest lower bounds ($K^{\wedge}_1 = K'_1 \wedge K''_1$ and $K^{\wedge}_2 = K'_2 \wedge K''_2$) and two least upper bounds ($K^{\vee}_1 = K'_1 \vee K''_1$ and $K^{\vee}_2 = K'_2 \vee K''_2$). By definition of the subkind relation $K^{\vee}_1 \to^{?' \wedge ?''} K^{\wedge}_2$ is then the greatest lower bound and $K^{\wedge}_1 \to^{?' \vee ?''} K^{\vee}_2$ the least upper bound. □

Proof of Lemma 5.13 on page 68: By induction on the combined length of derivation. The two parts for supremum are already covered by Lemma 5.5.

In the first part of the lemma, the only non-trivial case is where $\Gamma \vdash T \;+_A$ and $\Gamma \vdash T \;-_A$. The case for type variables is immediate, since $\Gamma \vdash A \;+_{A'}$ and $\Gamma \vdash A \;-_{A'}$ is derivable only if $A \neq A'$, and in this case we have $\Gamma \vdash A \;\circ_{A'}$. Likewise the cases where $A' \notin \mathit{fv}(T)$. The cases for Fun-types and All-types as the cases for arrow types are solved by straightforward induction and some calculation on the polarities.

For $\Gamma \vdash T_1\; T_2 \;+_A$ and $\Gamma \vdash T_1\; T_2 \;-_A$ we are given six polarities: p, q, and r on the one hand and p', q', and s' on the other, with $+ = p \vee q \times r$ and $- = p' \vee q' \times r'$:

$$\dfrac{\begin{array}{c} + = p \vee (q \times r) \\ \Gamma \vdash T_1\; p_A \qquad \Gamma \vdash T_1\; q \qquad \Gamma \vdash T_2\; r_A \end{array}}{\Gamma \vdash T_1\; T_2 \;+_A} \qquad \dfrac{\begin{array}{c} ?' = p' \vee (q' \times r') \\ \Gamma \vdash T_1\; p'_A \qquad \Gamma \vdash T_1\; q' \qquad \Gamma \vdash T_2\; r'_A \end{array}}{\Gamma \vdash T_1\; T_2 \;-_A}$$

By definition of $\vee$ we have $p \leq +$ and $p' \leq -$, so by induction $\Gamma \vdash T_1 \;\circ_{A'}$. Further we know $q \times r \leq +$ and $q \times r \leq -$. Assuming $(q, q') = (+, +)$, we get $r \leq +$ and $r' \leq -$, hence by induction $\Gamma \vdash T_2 \;\circ_A$ which implies $\Gamma \vdash T_1\; T_2 \;\circ_A$. If $(q, q') = (+, -)$, we get by induction on part 2 of the lemma $\Gamma \vdash T_1 \;\circ$, and again we can conclude $\Gamma \vdash T_1\; T_2 \;\circ_A$. The other combinations by similar calculations.

In the second part of the lemma, we proceed by induction on the combined length of derivation for $\Gamma \vdash T \in K_1$ and $\Gamma \vdash T \in K_2$. The case for $T = \mathit{Top}(\star)$ is immediate. For $T = A$, the case follows by Lemma 5.10 and Lemma 5.12. The cases for arrow types and universally quantified types are trivial, since they possess $\star$ as unique kind. The case for $T = \mathit{Fun}(A{:}K_1)T$ follows by induction.

Case: $T = U\; V$
Assume two kinds K_2 and K_4 with $\Gamma \vdash U\; V \in K_2$ and $\Gamma \vdash U\; V \in K_4$. We have to show that $\Gamma \vdash U\; V \in K$ when K is the infimum of K_2 and K_4. By the generation Lemma 5.8 $\Gamma \vdash U \in K_1 \to^{?} K_2$ and $\Gamma \vdash V \in K_1$ on the one hand, and $\Gamma \vdash U \in K_3 \to^{?'} K_4$ with $\Gamma \vdash V \in K_3$ on the other, each by subderivation.

By induction, we know $\Gamma \vdash U \in K'$ where K' is the infimum of $K_1 \rightarrow^{?} K_2$ and $K_3 \rightarrow^{?'} K_4$. By the properties of the subkind relation K' must also be an arrow-kind, say $K_5 \rightarrow^{?''} K_6$. By definition of infimum $K_5 \rightarrow^{?''} K_6 \leq K_1 \rightarrow^{?} K_2$ and $K_5 \rightarrow^{?''} K_6 \leq K_3 \rightarrow^{?'} K_4$; hence $K_1 \leq K_5$ and $K_3 \leq K_5$ and also $K_6 \leq K_2$ and $K_6 \leq K_4$. So by subsumption (Lemma 5.5) and K-ARROW-E we get $\Gamma \vdash U\ V \in K_6$. It is easy to check that K_6 must be the infimum of K_2 and K_4. The case for $\mathit{Top}(K' \rightarrow^{?} K'')$ is similar. □

Proof of soundness and completeness (Lemma 5.17 on page 70):
The algorithm's soundness is straightforward. For completeness we proceed by induction, using that all operations on polarities, including negation, are monotone.

For $\Gamma \vdash T \in K$ most cases are immediate. We show the one for arrow-elimination.

Case K-ARROW-E:

$$\frac{\Gamma \vdash S \in K_1 \rightarrow^{?} K_2 \qquad \Gamma \vdash T \in K_1}{\Gamma \vdash S\ T \in K_2}$$

By induction $\Gamma \vdash_{\mathcal{A}} S \in K$ and $\Gamma \vdash_{\mathcal{A}} T \in K_1''$, where $K \leq K_1 \rightarrow^{?} K_2$ and $K_1'' \leq K_1$. By definition of subkinds (rule K-SUB) K is of the form $K_1' \rightarrow^{?'} K_2'$ with $K_1 \leq K_1'$ and $K_2' \leq K_2$, as well as $?' \leq ?$. By transitivity of $\leq$ we get $K_1'' \leq K_1'$, and we can conclude by the arrow elimination rule of the algorithm:

$$\frac{\Gamma \vdash_{\mathcal{A}} S \in K_1' \rightarrow^{?'} K_2' \qquad \Gamma \vdash_{\mathcal{A}} T \in K_1'' \qquad K_1'' \leq K_1'}{\Gamma \vdash_{\mathcal{A}} S\ T \in K_2'}$$

□

Proof of Corollary 5.18 on page 71: Assume $\Gamma \vdash T\ ?_A$ respectively $\Gamma \vdash T \in K$. By the completeness part of Lemma 5.17 the algorithm gives back a statement $\Gamma \vdash_{\mathcal{A}} T\ ?'_A$ respectively $\Gamma \vdash T \in K'$ at least as good. This holds for all such statements. Thus by definition the algorithms gives back the minimal statement. □

Proof of Lemma 5.21 on page 72: All four parts by simple calculations using the properties of $\vee$, $\times$, and $\neg$. □

Proof of Lemma 5.22 on page 72: By simple calculations, using the properties of $\vee$ and $\times$, and $\neg$. □

Proof of weakening Lemma 5.23 on page 72: By induction on the depth of inference. □

Proof of the substitution Lemma 5.24 on page 72: The first part straightforwardly by induction. The second part by induction on the combined length of derivation of $\Gamma \vdash_{\mathcal{A}} T\ p_A$ and $\Gamma \vdash_{\mathcal{A}} T\ q_{A'}$. If $A \notin \mathit{fv}(T)$, the case follows by induction on the first part of the lemma. Since the algorithmic system is syntax directed, we list the remaining cases according to the syntactic structure of T.

Case: $T = A$
This means $p = +$ and $q = \circ$, and thus $s = +$. The statement $\Gamma' \vdash_{\mathcal{A}} A +_A$ is immediate.

Case: $T = A'$
Hence $p = \circ$ and $q = +$ (i.e. $s = r$). By assumption $\Gamma_1 \vdash_{\mathcal{A}} U\ r_A$ and by weakening $\Gamma' \vdash_{\mathcal{A}} U\ r_A$.

Case: $T = A''$
with $A \neq A'' \neq A'$. So $p = \circ$ and $q = \circ$, and we obtain immediately $\Gamma' \vdash_{\mathcal{A}} A''\ \circ_A$.

Case: $T = \mathit{Top}(K)$
By induction on the first or the second part of the lemma, depending of n the form of K.

Case: $T = T_1 \to T_2$
The algorithm generates as subgoals $\Gamma \vdash_{\mathcal{A}} T_1\ p'_A$ and $\Gamma \vdash_{\mathcal{A}} T_1\ q'_{A'}$ for the contravariant side, and $\Gamma \vdash_{\mathcal{A}} T_2\ p''_A$ and $\Gamma \vdash_{\mathcal{A}} T_2\ q''_{A'}$ for the co-variant, with $p = \neg p' \vee p''$ and $q = \neg q' \vee q''$. By induction $\Gamma' \vdash_{\mathcal{A}} [U/A]T_1\ s'_A$ and $\Gamma' \vdash_{\mathcal{A}} [U/A]T_2\ s''_{A'}$ with $s' = p' \vee (q' \times r)$ and $s'' = p'' \vee (q'' \times r)$. The algorithm gives back $\Gamma' \vdash_{\mathcal{A}} [U/A'](T_1 \to T_2)\ s_A$ with $s = \neg s' \vee s''$ and we can calculate, using the equations of Lemma 5.22:

$$\begin{aligned}
s &= \neg s' \vee s'' \\
&= \neg(p' \vee (q' \times r)) \vee (p'' \vee (q'' \times r)) \\
&= \neg p' \vee \neg(q' \times r) \vee p'' \vee (q'' \times r) \\
&= (\neg p' \vee p'') \vee \neg(q' \times r) \vee (q'' \times r) \\
&= (\neg p' \vee p'') \vee (\neg q \times r) \vee (q'' \times r) \\
&= p \vee ((\neg q' \vee q'') \times r) \\
&= p \vee (q \times r).
\end{aligned}$$

Case: $T = \mathit{All}(A''{\leq}T_1{:}K_1)T_2$
For both sides we have to distinguish according to the occurrence of the respective type variable in the upper bound of the All-type.

Subcase:

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1\ \circ_A \qquad \Gamma, A''{\leq}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ p_A}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\leq}T_1{:}K_1)T_2\ p_A} \qquad \frac{\Gamma \vdash_{\mathcal{A}} T_1\ \circ_{A'} \qquad \Gamma, A''{\leq}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ q_{A'}}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\leq}T_1{:}K_1)T_2\ q_{A'}}$$

By induction $\Gamma' \vdash_{\mathcal{A}} [U/A']T_1\ \circ_A$ (since $\circ = \circ \vee \circ \times r$). By induction on the second subgoals $\Gamma', A''{\leq}[U/A']T_1{:}K_1 \vdash_{\mathcal{A}} [U/A']T_2\ s_A$, where $s = p \vee (q \times r)$. Thus we can conclude the case with the corresponding rule for All-types.

Subcase:

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1 \circ_A \quad \Gamma, A''{\le}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ p_A}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\le}T_1{:}K_1)T_2\ p_A} \quad \frac{\Gamma \vdash_{\mathcal{A}} T_1\ q'_{A'} \quad q' \neq \circ \quad \Gamma, A''{\le}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ q''_{A'}}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\le}T_1{:}K_1)T_2 \pm_{A'}}$$

So the goal in this subcase is to derive $\Gamma' \vdash_{\mathcal{A}} ([U/A'](\mathit{All}(A''{\le}T_1{:}K_1)T_2)\ s_A$ where $s = p \vee (\pm \times r)$.

If $r = \circ$, then $s = p$, and we get by induction on the first subgoal $\Gamma' \vdash_{\mathcal{A}} [U/A']T_1\ \circ_A$, since $\circ \vee (s' \times \circ) = \circ$. Induction on the second subgoal and using $p \vee (q' \times \circ) = p$ yields $\Gamma', A''{\le}[U/A']T_1{:}K_1 \vdash_{\mathcal{A}} [U/A']T_2\ p_A$.

If $r \neq \circ$ we have $s = p \vee (\pm \times r) = \pm$. By induction on the first subgoal $\Gamma' \vdash_{\mathcal{A}} [U/A']T_1\ s'_A$, where $s' = \circ \vee (q' \times r) = q' \times r \neq \circ$, since neither q' nor r are constant. By induction on the second subgoal $\Gamma', A''{\le}[U/A']T_1{:}K_1 \vdash_{\mathcal{A}} [U/A']T_2\ s''_A$. Irrespective of s'', the corresponding rule for All-types derives $\Gamma', A''{\le}[U/A']T_1{:}K_1 \vdash_{\mathcal{A}} [U/A']T_2 \pm_A$.

Subcase:

$$\frac{\Gamma \vdash_{\mathcal{A}} T_1\ p'_A \quad p' \neq \circ \quad \Gamma, A''{\le}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ p''_A}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\le}T_1{:}K_1)T_2 \pm_A} \quad \frac{\Gamma \vdash_{\mathcal{A}} T_1 \circ_{A'} \quad \Gamma, A''{\le}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ q_{A'}}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\le}T_1{:}K_1)T_2\ q_{A'}}$$

We are to show $\Gamma' \vdash_{\mathcal{A}} ([U/A'](\mathit{All}(A''{\le}T_1{:}K_1)T_2) \pm_A$. By induction on the first subgoal $\Gamma' \vdash_{\mathcal{A}} [U/A]T_1\ s'_A$ with $s' = p' \vee (\circ \times r) = p' \neq \circ$, from which the case follows.

Subcase:

$$\frac{\overset{p' \neq \circ}{\Gamma \vdash_{\mathcal{A}} T_1\ p'_A \quad \Gamma, A''{\le}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ p''_A}}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\le}T_1{:}K_1)T_2 \pm_A} \quad \frac{\overset{q' \neq \circ}{\Gamma \vdash_{\mathcal{A}} T_1\ q'_{A'} \quad \Gamma, A''{\le}T_1{:}K_1 \vdash_{\mathcal{A}} T_2\ q''_{A'}}}{\Gamma \vdash_{\mathcal{A}} \mathit{All}(A''{\le}T_1{:}K_1)T_2 \pm_{A'}}$$

We have to show $\Gamma' \vdash_{\mathcal{A}} ([U/A'](\mathit{All}(A''{\le}T_1{:}K_1)T_2) \pm_A$. By induction on the first subgoal $\Gamma' \vdash_{\mathcal{A}} [U/A]T_1\ s'_A$ with $s' = p' \vee (q' \times r) \neq \circ$, from which the case follows.

Case: $T = \mathit{Top}(K)$
Immediate.

Case: $T = \mathit{Fun}(A''{:}K)T'$
By induction.

Case: $T = T_1\ T_2$
The aim is to show $\Gamma' \vdash_{\mathcal{A}} [U/A'](T_1\ T_2)\ s_A$ with $s = p \vee (q \times r)$. The corresponding rule of the algorithm generates $\Gamma \vdash_{\mathcal{A}} T_1\ p'_A$ and $\Gamma \vdash_{\mathcal{A}} T_1\ t$, and $\Gamma \vdash_{\mathcal{A}} T_2\ p''_A$; likewise

the subgoals $\Gamma \vdash_{\mathcal{A}} T_1\ {q'}_{A'}$ and $\Gamma \vdash_{\mathcal{A}} T_2\ {q'''}_{A'}$. Furthermore $p = p' \vee (t \times p''')$ and $q = q' \vee (t \times q''')$. By induction, we get $\Gamma' \vdash_{\mathcal{A}} [U/A']T_1\ {s'}_{A}$, and $\Gamma' \vdash_{\mathcal{A}} [U/A']T_1\ t$, and $\Gamma' \vdash_{\mathcal{A}} [U/A']T_2\ {s'''}_{A'}$, with $s' = p' \vee (q' \times r)$ and $s''' = p''' \vee (q''' \times r)$.

By the rule for application we get $\Gamma' \vdash_{\mathcal{A}} [U/A'](T_1\ T_2)\ s_A$, where $s = s' \vee (t \times s''')$. Thus we can calculate, using the properties of Lemma 5.22:

$$\begin{array}{rcl} s' \vee (t \times s''') & = & p' \vee (q' \times r) \vee (t \times (p''' \vee (q''' \times r))) \\ & = & p' \vee (q' \times r) \vee (t \times p''') \vee (t \times q''' \times r) \\ & = & p' \vee (t \times p''') \vee (q' \times r) \vee ((t \times q''') \times r) \\ & = & (p' \vee (t \times p''')) \vee ((q' \vee (t \times q''')) \times r) \\ & = & p \vee (q \times r) \\ & = & s. \end{array}$$

Finally the third part of the lemma, treating the rules for the kinding relation

Case K-TVAR:

$$\frac{\vdash_{\mathcal{A}} \Gamma\ ok}{\Gamma \vdash_{\mathcal{A}} A \in kind_{\Gamma} A}$$

The context Γ must be of the form $\Gamma_1, A {\leq} T{:}K', \Gamma_2$, i.e. $kind_{\Gamma} A = K'$. We distinguish, whether the two variables A and A' coincide or not.

If they do, we have by assumption $\Gamma_1 \vdash_{\mathcal{A}} U \in K'$ (i.e. $K = K'$). By induction on part 1 $\vdash \Gamma'\ ok$, thus by weakening (Lemma 5.23) also $\Gamma' \vdash_{\mathcal{A}} U \in K'$.

If $A \neq A'$, then $kind_{\Gamma'} A = kind_{\Gamma} A$, and the case follows by induction on part 1 and K-TVAR.

Case K-TOP:

$$\frac{\Gamma \vdash_{\mathcal{A}} Top(K_2) \in K_2'}{\Gamma \vdash_{\mathcal{A}} Top(K_1 \rightarrow^{?} K_2) \in K_1 \rightarrow^{\circ} K_2'}$$

By induction and K-TOP. The case for $Top(\star)$ by induction on the first part of the lemma.

Case K-ARROW-I?:

$$\frac{\Gamma, A''{:}K_1 \vdash_{\mathcal{A}} T\ ?'_{A''} \qquad \Gamma, A''{:}K_1 \vdash_{\mathcal{A}} T \in K_2'}{\Gamma \vdash_{\mathcal{A}} Fun(A''{:}K_1)T \in K_1 \rightarrow^{?'} K_2}$$

By induction $\Gamma', A''{:}K_1 \vdash_{\mathcal{A}} [U/A']T \in K_2'$ and, by induction on the second part of the lemma, $\Gamma', A''{:}K_1 \vdash_{\mathcal{A}} [U/A']T\ ?_{A''}$. Note that the fresh variable A'' cannot occur free in U, which means $\Gamma_1 \vdash_{\mathcal{A}} U \circ_{A''}$. Thus we can conclude by rule K-ARROW-I?:

$$\frac{\Gamma', A''{:}K_1 \vdash_{\mathcal{A}} [U/A']T\ ?_{A''} \qquad \Gamma', A''{:}K_1 \vdash_{\mathcal{A}} [U/A']T \in K_2'}{\Gamma' \vdash_{\mathcal{A}} Fun(A''{:}K_1)[U/A']T \in K_1' \rightarrow^{?'} K_2'}$$

Case K-Arrow-E-A:

$$\frac{\Gamma \vdash_{\mathcal{A}} S \in K_1 \to^{?} K_2 \qquad \Gamma \vdash_{\mathcal{A}} T \in K_1' \qquad K_1' \leq K_1}{\Gamma \vdash_{\mathcal{A}} S\ T \in K_2}$$

By induction we get $\Gamma' \vdash_{\mathcal{A}} [U/A']S \in K_1 \to^{?} K_2$ and $\Gamma' \vdash_{\mathcal{A}} [U/A']T \in K_1$. By K-Arrow-E-A we obtain $\Gamma' \vdash_{\mathcal{A}} [U/A'](S\ T) \in K_2$.

Case K-Arrow, K-All:
By induction.

That the same properties holds for non-algorithmic kinding, as well, is a direct consequence of soundness and completeness of the algorithm and monotonicity of the operations on polarities. □

Proof of subject reduction (Lemma 5.25 on page 73): We show the property not for arbitrary sequences of reduction steps but for one parallel step. Proceed by induction on the length of derivation in all three parts of the lemma.

The first part of the lemma by straightforward induction.

In the second part the rules for type variables and for *Top*-types are solved by induction over the first part of the lemma. The rules for type operators are also solved by induction. Likewise the ones for arrow-types and universally quantified types. We show one representative case:

Case :

$$\frac{\Gamma \vdash S_1 \circ_A \qquad \Gamma,\, A' {\leq} S_1{:}K_1 \vdash S_2\ ?_A}{\Gamma \vdash \mathit{All}(A' {\leq} S_1{:}K_1)S_2\ ?_A}$$

We are given $\Gamma \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \Gamma'$ and $\mathit{All}(A' {\leq} S_1{:}K_1)S_2 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \mathit{All}(A' {\leq} S_1'{:}K_1)S_2'$. By induction $\Gamma' \vdash S_1' \circ_A$ and, since $\Gamma,\, A' {\leq} S_1'{:}K_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \Gamma,\, A' {\leq} S_1{:}K_1$, also $\Gamma',\, A' {\leq} S_1'{:}K_1 \vdash S_2'\ ?_A$, so the result follows by the appropriate All-rule.

Finally the rules for application on the level of types. We distinguish according to the form of the outermost redex. (The case, where the outermost redex is not contracted, is solved by straightforward induction.)

Case :

$$\frac{\begin{array}{cc} \Gamma \vdash T\ ?'_A & ? = \circ \vee (\circ \times ?') \\ \Gamma \vdash \mathit{Top}(K_1 \to^{?} K_2) \circ_A & \Gamma \vdash \mathit{Top}(K_1 \to^{?} K_2) \circ \end{array}}{\Gamma \vdash \mathit{Top}(K_1 \to^{?} K_2)\ T\ ?_A}$$

We immediately get $? = \circ$, and after one step $\Gamma' \vdash \mathit{Top}(K_2) \circ_A$ holds: If $K_2 = \star$, this follows by induction on the first part of the lemma and K-Top; If $K_2 \neq \star$, then by induction on the first part.

Case :

$$\frac{\begin{array}{ccc} & ? = ?^1 \vee (?^2 \times ?^3) & \\ \Gamma \vdash (\mathit{Fun}(A'{:}K_1')S')\ ?^1{}_A & \Gamma \vdash (\mathit{Fun}(A'{:}K_1')S')\ ?^2 & \Gamma \vdash T\ ?^3{}_A \end{array}}{\Gamma \vdash (\mathit{Fun}(A'{:}K_1')S')\ T\ ?_A}$$

So assume $\mathit{Fun}(A'{:}K_1')S' \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \mathit{Fun}(A'{:}K_1')S''$ and $T \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} T''$. By induction we get $\Gamma' \vdash (\mathit{Fun}(A'{:}K_1')S'')\ ?^1{}_A$ and $\Gamma' \vdash \mathit{Fun}(A'{:}K_1')S''\ ?^2$ and $\Gamma' \vdash T'\ ?^3{}_A$. From the second statement we get from the generation Lemma 5.8 $\Gamma, A'{:}K_1' \vdash S'\ ?^2{}_{A'}$. From the kinding part of the generation lemma we further get $\Gamma \vdash \mathit{Fun}(A'{:}K_1)S' \in K_1' \rightarrow^{?^2} K$ for some kinds K_1' and K, as well as $\Gamma \vdash T \in K_1$ with $K_1 \leq K_1'$. Subsumption yields $\Gamma \vdash T \in K_1'$, so applying the substitution Lemma 5.24 gives $\Gamma' \vdash [T'/A']S''\ ?_A$.

Part three of the lemma by straightforward induction, using the generation lemma for kinding and preservation of kinding under reduction. □

C.2 Equivalence of types

Proof of Lemma 5.29 on page 74: By straightforward induction, using the formulation of the kinding system from Definition 5.3. □

Proof of Lemma 5.30 on page 74: By straightforward induction on the derivation of $\Gamma \vdash S \equiv T \in K$. The cases for E-Refl and E-Top follows by subsumption for kinds (Lemma 5.5). The cases for arrow- and for All-types are trivial as their kinds are unique. The case E-Abs by straightforward induction, likewise the ones for application, with the help of Lemma 5.5 for constant application. □

Proof of Lemma 5.32 on page 75: By induction on the depth of inference. The first part of the lemma by induction on the derivation of $\vdash \Gamma\ \mathit{ok}$. The case for the empty context is immediate, the one for term variables by straightforward induction. If $\Gamma = \Gamma_1, A{\leq}T{:}K$ we get by the generation lemma for contexts $\vdash \Gamma_1\ \mathit{ok}$ and $\Gamma_1 \vdash T \in K$ by subderivations. By definition of equivalence $\vdash \Gamma_1, A{\leq}T{:}K \equiv \Gamma'$ implies $\Gamma' = \Gamma_1', A{\leq}T'{:}K$ with $\vdash \Gamma_1 \equiv \Gamma_1'$ and $\Gamma_1 \vdash T \equiv T' \in K$, both by subderivation. Note that $A \notin \mathit{dom}(\Gamma)$ implies $A \notin \mathit{dom}(\Gamma')$. By induction we get $\vdash \Gamma_1'\ \mathit{ok}$, and induction on part 2 yields $\Gamma_1' \vdash T' \in K$. Hence $\vdash \Gamma_1', A{\leq}T'{:}K\ \mathit{ok}$.

Part 2 of the lemma by induction on the depth of inference of $\Gamma \vdash T \in K$.

Case K-TVar-Sub:
$$\frac{\mathit{kind}_\Gamma A \leq K \qquad \vdash \Gamma\ \mathit{ok}}{\Gamma \vdash A \in K}$$

The only equivalence rule applicable for type variables is reflexivity, i.e., by the kinding subgoal of E-Refl we are given $\Gamma \vdash A \in K'$. From this we get by the generation lemma for kinding $\mathit{kind}_{\Gamma'} A \leq K'$. Thus the case follows by induction on part 1 of the lemma and the observation that $\mathit{kind}_{\Gamma'} A = \mathit{kind}_\Gamma A$.

Case K-Top⋆:
$$\frac{\vdash \Gamma\ \mathit{ok}}{\Gamma \vdash \mathit{Top}(\star) \in \star}$$

We get $T' = T = \mathit{Top}(\star)$, so the case follows by induction on part 1 of the lemma.

Case K-Top-Sub:
$$\frac{K_1 \leq K_1'' \qquad \Gamma \vdash \mathit{Top}(K_2'') \in K_2}{\Gamma \vdash \mathit{Top}(K_1'' \rightarrow^{?''} K_2'') \in K_1 \rightarrow^{?} K_2}$$

From $\Gamma \vdash \mathit{Top}(K_1'' \rightarrow^{?''} K_2'') \equiv T' \in K_1' \rightarrow^{?'} K_2'$ we obtain by rule E-Top $T' = \mathit{Top}(K_1''' \rightarrow^{?'''} K_2''')$, where $\Gamma \vdash \mathit{Top}(K_1'' \rightarrow^{?''} K_2'') \in K_1' \rightarrow^{?'} K_2'$ and $\mathit{Top}(K_1''' \rightarrow^{?'''} K_2''') \in K_1' \rightarrow^{?'} K_2'$. By the generation Lemma 5.8 for kinds $\Gamma \vdash \mathit{Top}(K_2'') \in K_2'$ and $\Gamma \vdash \mathit{Top}(K_2''') \in K_2'$ with $K_1' \leq K_1''$ and $K_1' \leq K_1'''$. By E-Top we thus know $\Gamma \vdash \mathit{Top}(K_2'') \equiv \mathit{Top}(K_2''') \in K_2'$. Hence by induction $\Gamma' \vdash \mathit{Top}(K_2''') \in K_2'$, and the result follows by K-Top-Sub:

$$\frac{K_1' \leq K_1''' \qquad \Gamma' \vdash \mathit{Top}(K_2''') \in K_2'}{\Gamma' \vdash \mathit{Top}(K_1''' \rightarrow^{?'''} K_2''') \in K_1' \rightarrow^{?'} K_2'}$$

Case K-Arrow-I-Sub:
$$\frac{\begin{array}{c} K_1 \leq K_1' \\ \Gamma, A{:}K_1' \vdash T\ ?_A \qquad \Gamma, A{:}K_1' \vdash T \in K_2 \end{array}}{\Gamma \vdash \mathit{Fun}(A{:}K_1')T \in K_1 \rightarrow^{?} K_2}$$

By induction on part 2 and 4 of the lemma.

Case K-Arrow-E:
$$\frac{\Gamma \vdash T_1 \in K_1 \rightarrow^{?} K_2 \qquad \Gamma \vdash T_2 \in K_1}{\Gamma \vdash T_1\ T_2 \in K_2}$$

By definition of equivalence, $\Gamma \vdash T_1\ T_2 \equiv T' \in K_2'$ implies $T = T_1'\ T_2'$ with $\Gamma \vdash T_1 \equiv T_1' \in K_1' \rightarrow^{?'} K_2'$ and $\Gamma \vdash T_2 \equiv T_2' \in K_1'$. By induction $\Gamma' \vdash T_2' \in K_1'$ and $\Gamma' \vdash T_1' \in K_1' \rightarrow^{?'} K_2'$, and the case follows by K-Arrow-E.

Case K-Arrow:
$$\frac{\Gamma \vdash T_1 \in \star \qquad \Gamma \vdash T_2 \in \star}{\Gamma \vdash T_1 \rightarrow T_2 \in \star}$$

By E-Arrow, $T' = T_1' \rightarrow T_2'$ with $\Gamma \vdash T_1 \equiv T_1' \in \star$ and $\Gamma \vdash T_2 \equiv T_2' \in \star$ and the result follows by induction.

Case K-All:
$$\frac{\Gamma, A{\leq}T_1{:}K_1 \vdash T_2 \in \star}{\Gamma \vdash \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

We are given $T' = \mathit{All}(A{\leq}T_1'{:}K_1)T_2'$ with $\Gamma \vdash T_1 \equiv T_1' \in K_1$ and $\Gamma, A{\leq}T_1{:}K_1 \vdash T_2 \equiv T_2' \in \star$. By the definition of equivalence for contexts and using $\vdash \Gamma \equiv \Gamma'$ we get $\vdash \Gamma, A{\leq}T_1{:}K_1 \equiv \Gamma', A{\leq}T_1'{:}K_1$, so by induction and K-All we conclude:

$$\frac{\Gamma', A{\leq}T_1'{:}K_1 \vdash T_2' \in \star}{\Gamma' \vdash \mathit{All}(A{\leq}T_1'{:}K_1)T_2' \in \star}$$

The cases for $\Gamma \vdash T\ ?$ and for variable occurrence $\Gamma \vdash T\ ?_A$ also by straightforward induction. □

Proof of Lemma 5.33 on page 75: By induction on the length of derivation, using the corresponding weakening Lemma 5.2 for kinding. $\square$

Proof of symmetry and transitivity (Lemma 5.34 on page 75): By induction on the length of derivation. The only interesting case is the one for *All*-types:

Case E-ALL:

$$\frac{\Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1}{\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \equiv \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

Since S_1 and T_1 are well-kinded in Γ, the induction hypothesis applies, yielding $\Gamma' \vdash T_1 \equiv S_1 \in K_1$. Thus $\vdash \Gamma, A{\leq}S_1{:}K_1 \equiv \Gamma', A{\leq}T_1{:}K_1$. Again by induction $\Gamma', A{\leq}T_1{:}K_1 \vdash T_2 \equiv S_2 \in \star$, and the case finishes with rule E-ALL.

In the part for transitivity also by induction. I only show the case where both derivations end with an instance of E-ALL, the only rule which is not symmetric.

Case E-ALL:

$$\frac{\begin{array}{c}\Gamma \vdash S_1 \equiv U_1 \in K_1 \\ \Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \equiv U_2 \in \star\end{array}}{\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \equiv \mathit{All}(A{\leq}U_1{:}K_1)U_2 \in \star} \quad \frac{\begin{array}{c}\Gamma' \vdash U_1 \equiv T_1 \in K_1 \\ \Gamma', A{\leq}U_1{:}K_1 \vdash U_2 \equiv T_2 \in \star\end{array}}{\Gamma' \vdash \mathit{All}(A{\leq}U_1{:}K_1)U_2 \equiv \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

By induction $\Gamma \vdash S_1 \equiv T_1 \in K_1$. Since by definition of equivalence for contexts $\vdash \Gamma, A{\leq}S_1{:}K_1 \equiv \Gamma', A{\leq}U_1{:}K_1$, by induction $\Gamma', A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star$, and the case follows with E-ALL. $\square$

Proof of Corollary 5.35 on page 75: In the first part, assume $\Gamma' \vdash_{\mathcal{A}} T'\ ?'$, with $?' \leq ?$. By symmetry, the corresponding lemma for non-minimal kinding (part 3 of Lemma 5.32), and soundness of the kinding algorithm (Lemma 5.17) $\Gamma' \vdash T \in ?'$. Since $?$ was assumed minimal for T in Γ, we get $? \leq ?'$, and thus $? = ?'$. The second part is similar. $\square$

Proof of the substitution Lemma 5.36 on page 76: Let Γ stand for the context $\Gamma_1, A{:}K', \Gamma_2$ and Γ' abbreviate $\Gamma_1, [U_1/A]\Gamma_2$. Proceed by induction on the length of derivation of $\Gamma \vdash S \in K$.

Case: $S = A'$

If $A = A'$, we have by assumption $\Gamma_1 \vdash U_1 \equiv U_2 \in K$. By the generation lemma for kinds, $\mathit{kind}_\Gamma A = K' \leq K$, thus the case follows by weakening for equivalence (Lemma 5.33) and Lemma 5.30. If $A \neq A'$ the case is immediate by reflexivity, the generation lemma for kinds, and subsumption for kinding.

Case: $\Gamma \vdash All(A'{\leq}S_1{:}K_1)S_2 \in \star$
By the generation lemma for kinds Γ, $A'{\leq}S_1{:}K_1 \vdash S_2 \in \star$ and further $\Gamma \vdash S_1 \in K_1$, both by subderivations. By induction $\Gamma' \vdash [U_1/A]S_1 \equiv [U_2/A]S_1 \in K_1$ and $\Gamma', A'{\leq}[U_1/A]S_1{:}K_1 \vdash [U_1/A]S_2 \equiv [U_2/A]S_2 \in \star$. Thus we conclude:

$$\frac{\Gamma' \vdash [U_1/A]S_1 \equiv [U_2/A]S_1 \in K_1 \qquad \Gamma', A'{\leq}[U_1/A]S_1{:}K_1 \vdash [U_1/A]S_2 \equiv [U_2/A]S_2 \in \star}{\Gamma' \vdash [U_1/A](All(A'{\leq}S_1{:}K_1)S_2) \equiv [U_2/A](All(A'{\leq}S_1{:}K_1)T_2) \in \star}$$

□

Proof of the substitution Lemma 5.37 on page 76: By induction over the length of derivation, using the generation lemma for kinds, the preservation of kinding under substitution, and the above substitution Lemma 5.36 in the case for E-Refl. □

Proof of the constant substitution Lemma 5.39 on page 77: By structural induction on the structure of type S. For $S = A'$, the assumption $\Gamma \vdash S \circ_A$ implies $A' \neq A$. Thus the substitution has no effect and the case is immediate by reflexivity of equivalence, and preservation of kinding under substitution. The cases for arrow-types, All-types, and type operators follow by the generation lemma for kinds, induction, and the corresponding rules for equivalence. We only show the one for type applications:

Case: $S = S_1\ S_2$
By definition of equivalence, $T = T_1\ T_2$ with $\Gamma \vdash S_1 \equiv T_1 \in K_1 \rightarrow^? K$ for some kind K_1 and some polarity ?. By Lemma 5.29 and the generation lemma for kinds we get $\Gamma \vdash S \circ_A$. Hence by induction $\Gamma' \vdash [U_1/A]S_1 \equiv [U_2/A]T_1 \in K_1 \rightarrow^? K$.

To treat the arguments S_2 and T_2, we distinguish according to the minimal polarity of S_1 as type operator. If $\Gamma \vdash_{\mathcal{A}} S_1 \circ$, we know by the subgoals of rule E-App∘ that $\Gamma \vdash S_2 \in K_1$ and $\Gamma \vdash T_2 \in K_1$. By preservation of kinding under substitution (Lemma 5.24) $\Gamma' \vdash [U_1/A]S_2 \in K_1$ and $\Gamma' \vdash [U_2/A]T_2 \in K_1$, and the result follows by E-App∘:

$$\frac{\begin{array}{c}\Gamma' \vdash [U_1/A]S_2 \in K_1 \qquad \Gamma' \vdash [U_2/A]T_2 \in K_1 \\ \Gamma' \vdash [U_1/A]S_1 \equiv [U_2/A]T_1 \in K_1 \rightarrow K \qquad \Gamma' \vdash_{\mathcal{A}} [U_1/A]S_1 \circ\end{array}}{\Gamma' \vdash [U_1/A](S_1\ S_2) \equiv [U_2/A](T_1\ T_2) \in K}$$

If $\Gamma \vdash_{\mathcal{A}} S_1\ ?$ with $? \neq \circ$, we get by the generation Lemma 5.21 for minimal kinds that $\Gamma \vdash S_2 \circ_A$. By the appropriate application rule for equivalence we know $\Gamma \vdash S_2 \equiv T_2 \in K_1$, hence by induction $\Gamma' \vdash [U_1/A]S_2 \equiv [U_2/A]T_2$, and the case follows by the corresponding application rule. □

Proof of Lemma 5.40 on page 77: Part 1 by induction on the length of derivation, using the previous lemmas. The case for E-Refl follows from preservation of kinding under reduction and E-Refl, the one for E-Top vacuously true. The case for E-Abs is solved by straightforward induction and preservation of kinding under reduction.

Case E-Arrow:

$$\frac{\Gamma \vdash T_1 \equiv S_1 \in \star \qquad \Gamma \vdash S_2 \equiv T_2 \in \star}{\Gamma \vdash S_1 \to S_2 \equiv T_1 \to T_2 \in \star}$$

If $S_1 \to S_2 \longrightarrow_{\beta\top} S_1' \to S_2$ with $S_1 \longrightarrow_{\beta\top} S_1'$, we get by subject reduction and induction a type T_1' with $T_1 \longrightarrow_{\beta\top} T_1'$ and $\Gamma \vdash S_1 \equiv T_1 \in \star$ and we conclude:

$$\frac{\Gamma \vdash S_1' \equiv T_1 \in \star \qquad \Gamma \vdash S_2 \equiv T_2 \in \star}{\Gamma \vdash S_1' \to S_2 \equiv T_1' \to T_2 \in \star}$$

The case where $S_1 \to S_2 \longrightarrow_{\beta\top} S_1 \to S_2'$ is symmetric.

Case E-All:

$$\frac{\Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1}{\Gamma \vdash All(A{\leq}S_1{:}K_1)S_2 \equiv All(A{\leq}T_1{:}K_1)T_2 \in \star}$$

We have to distinguish where the reduction step takes place:

Subcase: $S' = All(A{\leq}S_1'{:}K_1)S_2$
with $S_1 \longrightarrow_{\beta\top} S_1'$. By well-kinded subderivations $\Gamma \vdash S_1 \in K_1$, by subject reduction for kinding also $\Gamma \vdash S_1' \in K_1$. Thus the induction hypothesis applies, yielding a type T_1' with

$$\begin{tikzcd}
S_1 \arrow[r, equal, "\Gamma", "K_1"'] \arrow[d, "\beta\top"'] & T_1 \arrow[d, "\beta\top"', "*"] \\
S_1' \arrow[r, equal, "\Gamma", "K_1"'] & T_1'.
\end{tikzcd}$$

With the help of Lemma 5.32 we get $\Gamma, A{\leq}S_1'{:}K_1 \vdash S_2 \equiv T_2 \in \star$ and the case follows by the equality rule for *All*-types.

Subcase: $S' = All(A{\leq}S_1{:}K_1)S_2'$
with $S_2 \longrightarrow_{\beta\top} S_2'$. By well-kindedness of subderivation and induction we get a type T_2' such that

$$\begin{tikzcd}
S_2 \arrow[r, equal, "{\Gamma,\, A{\leq}S_1{:}K_1}", "\star"'] \arrow[d, "\beta\top"'] & T_2 \arrow[d, "\beta\top"', "*"] \\
S_2' \arrow[r, equal, "{\Gamma,\, A{\leq}S_1{:}K_1}", "\star"'] & T_2',
\end{tikzcd}$$

and the case follows again by the rule for universally quantified types.

Case E-App+: $$\frac{\Gamma \vdash_{\mathcal{A}} S_1 + \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \rightarrow K_2 \qquad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2}$$

We have to distinguish where the reduction step takes place.

Subcase: $S_1\ S_2 \longrightarrow_{\beta\top} S_1'\ S_2$

By induction, using subject reduction for minimal kinds. In case the minimal kind of S_1' is constant, we can use Lemma 5.29 and E-App∘ to obtain the result. The case where $S_1\ S_2 \longrightarrow_{\beta\top} S_1\ S_2'$ is symmetric.

Subcase: $(\mathit{Fun}(A{:}K_1')S_1')\ S_2 \longrightarrow_{\beta} [S_2/A]S_1'$

$\Gamma \vdash \mathit{Fun}(A{:}K_1')S_1' \equiv T_1 \in K_1 \rightarrow K_2$ implies $T_1 = \mathit{Fun}(A{:}K_1')T_1'$ where $\Gamma,\ A{:}K_1' \vdash S_1' \equiv T_1' \in K_2$ and $K_1 \leq K_1'$. From $\Gamma \vdash S_2 \equiv T_2 \in K_1$ we get by Lemma 5.30 $\Gamma \vdash S_2 \equiv T_2 \in K_1'$, so the result follows with the substitution Lemma 5.37:

$$\begin{tikzcd}
(\mathit{Fun}(A{:}K_1')S_1')S_2 \arrow[r, equal, "\Gamma", "K_2"'] \arrow[d, "\beta"'] & (\mathit{Fun}(A{:}K_1')T_1')T_2 \arrow[d, "\beta"'] \\
{[S_2/A]S_1'} \arrow[r, equal, "\Gamma", "K_2"'] & {[T_2/A]T_1'.}
\end{tikzcd}$$

Subcase: $\mathit{Top}(K_1' \rightarrow^{?'} K_2')\ S_2 \longrightarrow_{\top} \mathit{Top}(K_2')$

$\Gamma \vdash \mathit{Top}(K_1' \rightarrow^{?'} K_2') \equiv T_1$ implies $T_1 = \mathit{Top}(K_1'' \rightarrow^{?''} K_2'')$ with $\Gamma \vdash \mathit{Top}(K_1' \rightarrow^{?'} K_2') \equiv \mathit{Top}(K_1'' \rightarrow^{?''} K_2'') \in K_1 \rightarrow^{?} K_2$ and $\Gamma \vdash S_2 \equiv T_2 \in K_1$. But now $\Gamma \vdash_{\mathcal{A}} \mathit{Top}(K_1' \rightarrow^{?'} K_2') \circ$ and $\Gamma \vdash_{\mathcal{A}} \mathit{Top}(K_1' \rightarrow^{?'} K_2') \circ$, which means that this case cannot occur as subcase of the monotone application rule.

Case E-App∘: $$\frac{\Gamma \vdash_{\mathcal{A}} S_1 \circ \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \rightarrow K_2 \qquad \Gamma \vdash S_2, T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2}$$

Similar. In case the reduction step takes place inside S_2, i.e., $S_1\ S_2 \longrightarrow_{\beta\top} S_1\ S_2'$, we choose $T' = T$, since $\Gamma \vdash S_1\ S_2' \equiv T_1\ T_2 \in K_2$.[1] Unlike in the case for E-App+, we have to consider *Top* as constant type operator:

[1]This case is the reason, why we cannot have the stronger property that $T \longrightarrow_{\beta\top} T'$ in exactly one step.

Subcase: $\mathit{Top}(K_1' \to^{?'} K_2')\ S_2 \longrightarrow_{\top} \mathit{Top}(K_2')$
So we are given $T_1 = \mathit{Top}(K_1'' \to^{?''} K_2'')$ with $\Gamma \vdash \mathit{Top}(K_1' \to^{?'} K_2') \equiv \mathit{Top}(K_1'' \to^{?''} K_2'') \in K_1 \to^{?} K_2$. Thus by Lemma 5.29 and the generation lemma for kinding (Lemma 5.8) $\Gamma \vdash \mathit{Top}(K_2') \in K_2$ and likewise $\Gamma \vdash \mathit{Top}(K_2'') \in K_2$. Rule E-Top hence justifies $\Gamma \vdash \mathit{Top}(K_2') \equiv \mathit{Top}(K_2'') \in K_2$ and we can conclude:

$$\begin{tikzcd}
\mathit{Top}(K_1' \to^{?'} K_2')\ S_2 \arrow[r, equal, "\Gamma", "K_2"'] \arrow[d, "\top"'] & \mathit{Top}(K_1'' \to^{?''} K_2'')\ T_2 \arrow[d, "\top"'] \\
\mathit{Top}(K_2') \arrow[r, equal, "\Gamma", "K_2"'] & \mathit{Top}(K_2'').
\end{tikzcd}$$

For normalization we use part 1 to obtain from $\Gamma \vdash S \equiv T \in K$ and $S \longrightarrow^{!}_{\beta\top} S^{!}$ a type T' such that $T \longrightarrow^{*}_{\beta\top} T'$ and $\Gamma \vdash S^{!} \equiv T' \in K'$. Type T' is not necessarily in normal form, but we have $\Gamma \vdash S^{!} \equiv T'^{!} \in K$, from which the result follows.

The last implication is a consequence of the following lemma:

> If $\Gamma \vdash S^{!} \equiv T' \in K$ with $S^{!}$ in normal form and $T' \longrightarrow_{\beta\top} T''$, then $\Gamma \vdash S^{!} \equiv T'' \in K$.

This can be proven by induction on the syntactic structure of $S^{!}$, using symmetry of equivalence for E-Arrow and E-App−. □

Proof of Lemma 5.41 on page 77: The definition of promotion implies $S = A\ S_1 \ldots S_n$ and $S' = \Gamma(A)\ S_1 \ldots S_n$. Proceed by induction on $\Gamma \vdash A\ S_1 \ldots S_n \equiv T \in K$. □

C.3 Properties of the reducing system

Proof of Lemma 5.45 on page 79: By the definition of promotion, only types of the form $A\ T_1 \ldots T_n$ can be promoted. Proceed by induction on n.

For $n = 0$, we are given $\Gamma \vdash A \in K$. By the generation Lemma for kinds (Lemma 5.8) $\mathit{kind}_{\Gamma} A \leq K$, and by generation for contexts (Lemma 5.4) $\vdash \Gamma\ \mathit{ok}$. The context must be of the form $\Gamma_1, A{\leq}S{:}K', \Gamma_2$ (i.e. $K' = \mathit{kind}_{\Gamma} A$), from which by the same generation lemma $\Gamma_1 \vdash S \in K'$. Using weakening for kinds (Lemma 5.2) we arrive at $\Gamma \vdash S \in K'$, and finally by subsumption (Lemma 5.5) $\Gamma \vdash S \in K$.

For $n > 0$ by the generation Lemma 5.8 $\Gamma \vdash A\ T_1 \ldots T_{n-1} \in K_1 \to^{?} K$ and $\Gamma \vdash T_n \in K_1$ for some kind K_1. By induction $\Gamma \vdash \Gamma(A)\ T_1 \ldots T_{n-1} \in K_1 \to^{?} K_2$, and the result follows with K-Arrow-E. □

Proof of Lemma 5.47 on page 79: The assumption $\Gamma \vdash T \in K$ implies with Lemma 5.4 that $\vdash \Gamma$ *ok*. By definition of promotion, only types of the form $A'\, T_1 \ldots T_n$ can be promoted. Proceed by induction on n.

For $n = 0$ we have $T' = \Gamma(A')$, with A' not necessarily different from A. The context Γ must be of the form $\Gamma_1,\ A'{\leq}T'{:}K',\ \Gamma_2$. By Definition 5.38 $\Gamma_1 \vdash T' \circ_A$ which, by weakening for kinding (Lemma 5.2) and subsumption (Lemma 5.5) means that all polarities are derivable for A.

For $n > 0$ we get by the generation Lemma 5.8 $\Gamma \vdash A'\ T_1 \ldots T_{n-1} \in K_1 \to^? K$ for some kind K_1. By induction, the statement $\Gamma \vdash A'\ T_1 \ldots T_{n-1}\ ?'_A$ entails $\Gamma \vdash \Gamma(A')\ T_1 \ldots T_{n-1}\ ?'_A$ for all polarities $?'$. Thus the result follows by the preservation of kinding under promotion (Lemma 5.45) and the polarity-rule for applications. $\square$

Proof of the expansion Lemma 5.48 on page 79: By straightforward induction and inspection of the rules of the reducing system from Definition 5.43.
$\square$

Proof of Lemma 5.49 on page 81: By induction over the length of derivations, using preservation of kinding under reduction and under promotion (Lemma 5.25 and Lemma 5.45), and the generation Lemma 5.8 for kinds at various places. We show some illustrative cases.

Case R-Refl:

$$\frac{S \longrightarrow^*_{\beta\top} U \qquad T \longrightarrow^*_{\beta\top} U \qquad \Gamma \vdash U \in K}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K}$$

Since S and T are well-kinded in Γ, so is U by subject reduction for kinding. By the same lemma further $\Gamma \vdash U^! \in K$.

Case R-Promote:

$$\frac{S \longrightarrow^*_{\beta\top} U \uparrow_\Gamma U' \qquad \Gamma \vdash U \in K \qquad \Gamma \vdash_{\mathcal{R}} U' \leq T \in K}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K}$$

By preservation of kinding under reduction and uniqueness of normal forms $\Gamma \vdash S^! \in K$. For the type on the right-hand side by induction $\Gamma \vdash T^! \in K$.

Case R-Abs:

$$\frac{\begin{array}{c} S \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')S' \qquad T \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')T' \\ K_1 \leq K_1' \qquad \Gamma,\ A{:}K_1' \vdash_{\mathcal{R}} S' \leq T' \in K_2 \qquad \Gamma,\ A{:}K_1' \vdash S', T'\ ?_A \end{array}}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K_1 \to^? K_2}$$

Since S and T are well-kinded in Γ, so are by subject reduction $\mathit{Fun}(A{:}K_1')S'$ and $\mathit{Fun}(A{:}K_1')T'$. By the generation Lemma 5.8 S' and T' are well-kinded in $\Gamma,\ A{:}K_1'$. By induction $\Gamma,\ A{:}K_1' \vdash S'^! \in K_2$ and $\Gamma,\ A{:}K_1' \vdash T'^! \in K_2$. Further by rule K-Arrow-I-Sub and the preservation of variable occurrence under reduction from

Lemma 5.25:

$$\frac{K_1 \leq K_1' \qquad \Gamma, A{:}K_1' \vdash S'^{!}\ ?_A \qquad \Gamma, A{:}K_1' \vdash S'^{!} \in K_2}{\Gamma \vdash \mathit{Fun}(A{:}K_1')S'^{!} \in K_1 \rightarrow^{?} K_2}$$

Similarly for $\mathit{Fun}(A{:}K_1')T'^{!}$. □

Proof of Lemma 5.50 on page 81: By induction on the length of derivation, using well-kindedness of subderivations and the corresponding weakening lemmas for kinding and equivalence (Lemma 5.2 and 5.33). □

Proof of maximality of *Top* (Lemma 5.51 on page 81): By induction on the length of derivation, using well-kindedness of subderivations. In the cut-free system, only the rules R-Refl, R-Top, or one of the application rules are applicable. Reflexivity and the *Top*-rule are immediate. As we choose the application rule according to the best polarity of one of the operators, only one of the two rules for constant application applies, for example:

Case R-App$\circ_l$:

$$\frac{\begin{array}{c} \mathit{Top}(K)\ S_1 \ldots S_n \longrightarrow^{*}_{\beta\top} \mathit{Top}(K_1' \rightarrow^{?'} K_2')\ U_1 \ldots U_m \quad T \longrightarrow^{*}_{\beta\top} T_1\ T_2 \\ \Gamma \vdash_{\mathcal{A}} \mathit{Top}(K_1' \rightarrow^{?'} K_2')\ U_1 \ldots U_{m-1} \circ \\ \Gamma \vdash_{\mathcal{C}} \mathit{Top}(K_1' \rightarrow^{?'} K_2')\ U_1 \ldots U_{m-1} \leq T_1 \in K_1'' \rightarrow K_2 \end{array}}{\Gamma \vdash_{\mathcal{C}} \mathit{Top}(K)\ S_1 \ldots S_n \leq T \in K_2}$$

with $m > 0$. The form of the left-hand side after reduction is justified by the properties of the reduction relation, preservation of kinding under reduction and the generation lemma for kinds. That type $\mathit{Top}(K_1' \rightarrow^{?'} K_2')\ U_1 \ldots U_{m-1}$ is a constant operator follows also by preservation of kinding under reduction and the kinding rule for maximal types. Hence by induction $T_1 \longrightarrow^{*}_{\beta\top} \mathit{Top}(K'')$, i.e., $T_1 \longrightarrow^{!}_{\beta\top} \mathit{Top}(K'')$, for some kind K''. By well-kindedness of subderivations (Lemma 5.49) $\Gamma \vdash \mathit{Top}(K'') \in K_1'' \rightarrow^{?''} K_2$, with K'' being an arrow-kind $K_3 \rightarrow^{?'''} K_4$ and $\mathit{Top}(K_3 \rightarrow^{?'''} K_4)\ U_m$ well-kinded in Γ. So by one last $\top$-step $\mathit{Top}(K_3 \rightarrow^{?''} K_4)U_m \longrightarrow_{\top} \mathit{Top}(K_4)$. □

Proof of Lemma 5.52 on page 81: By induction on the length of derivation, preservation of kinding under reduction, and subsumption on the level of kinds.

The cases for reflexivity and for the *Top*-types follow from subsumption on the level of kinds. R-Trans by straightforward induction. The rule R-Promote by induction, using preservation of kinding under reduction and the promotion rule. The rules for arrow types and for universally quantified types are trivial, as their kind is unique by Lemma 5.8.

Case R-App$\pm_l$:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{A}} S_1 \pm \qquad \Gamma \vdash_{\mathcal{R}} S_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K_2}$$

Assume a kind K_2' with $K_2 \leq K_2'$. By K-Sub, $K_1 \to K_2 \leq K_1 \to K_2'$, and the case follows by well-kindedness of subderivations and induction.

Case R-App$+_l$:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{A}} S_1 + \qquad \Gamma \vdash_{\mathcal{R}} S_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash_{\mathcal{R}} S_2 \leq T_2 \in K_1}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K_2}$$

Assume again a kind K_2' with $K_2 \leq K_2'$. Hence $K_1 \to K_2 \leq K_1 \to K_2'$ and the case follows by induction and by R-App$+_l$.

Case R-Abs:

$$\frac{S \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')S' \qquad T \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')T' \qquad K_1 \leq K_1' \qquad \Gamma, A{:}K_1' \vdash_{\mathcal{R}} S' \leq T' \in K_2 \qquad \Gamma, A{:}K_1' \vdash S', T'\ ?_A}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K_1 \to^{?} K_2}$$

Assume a kind K'' with $K_1 \to^? K_2 \leq K''$, i.e., $K'' = K_1'' \to^{?''} K_2''$ where $K_1'' \leq K_1$ and $K_2 \leq K_2''$. By induction $\Gamma, A{:}K_1' \vdash_{\mathcal{R}} S' \leq T' \in K_2''$; by transitivity of subkinding $K_1'' \leq K_1'$, and the case follows by R-Abs. □

Lemma C.1 If $\Gamma \vdash T \in K_1 \to^? K_2$ and $\Gamma \vdash T \in K_1' \to^{?'} K_2'$, then $\Gamma \vdash T \in K_1' \to^? K_2$.

Proof of Lemma C.1: By induction on the length of derivation. The cases K-Top$\star$, K-Arrow, and K-All cannot occur. The case for $T = \mathit{Top}(K_1'' \to^{?''} K_2''$, by the generation lemma for kinds and K-Top. Similarly for $T = \mathit{Fun}(A{:}K_1'')T'$ with K-Arrow-I-Sub. The case for application finally by induction and K-Arrow-E. □

Proof of Lemma 5.53 on page 81: By induction on the length of derivation. R-Refl and R-Top by preservation of kinding under reduction. R-Promote by preservation of kinding under reduction and promotion, and by induction. R-Arrow and R-All are trivial, as their kind $\star$ is unique.

Case R-Abs:

$$\frac{S \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1'')S' \qquad T \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1'')T' \qquad K_1 \leq K_1'' \qquad \Gamma, A{:}K_1'' \vdash_{\mathcal{C}} S' \leq T' \in K_2 \qquad \Gamma, A{:}K_1'' \vdash S', T'\ ?_A}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_1 \to^{?} K_2}$$

By preservation of kinding under reduction, $\Gamma \vdash \mathit{Fun}(A{:}K_1'')S' \in K'$ and $\Gamma \vdash \mathit{Fun}(A{:}K_1'')T' \in K'$. By the generation lemma for kinding $K' = K_1' \to^{?'} K_2'$, with $K_1' \leq K_1''$, with $\Gamma, A{:}K_1'' \vdash S' \in K_2'$ and $\Gamma, A{:}K_1'' \vdash T' \in K_2'$, and with $\Gamma, A{:}K_1'' \vdash S'\ ?'_A$ and $\Gamma, A{:}K_1'' \vdash T'\ ?'_A$. Thus, by induction $\Gamma, A{:}K_1'' \vdash_{\mathcal{C}} S' \leq T' \in K_2'$, and the result follows by R-Abs and the expansion lemma.

Case R-App$+_l$:

$$\frac{\begin{array}{c} S \longrightarrow^{*}_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^{*}_{\beta\top} T_1\ T_2 \\ \Gamma \vdash_{\mathcal{A}} S_1\ + \\ \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \rightarrow K_2 \qquad \Gamma \vdash_{\mathcal{C}} S_2 \leq T_2 \in K_1 \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_2}$$

So, assume $\Gamma \vdash S \in K_2'$ and $\Gamma \vdash T \in K_2'$. By preservation of kinding under reduction, $\Gamma \vdash S_1\ S_2 \in K_2'$ and $\Gamma \vdash T_1\ T_2 \in K_2'$. By the generation lemma for kinding, $\Gamma \vdash S_1 \in K_1' \rightarrow^{?'} K_2'$ and $\Gamma \vdash S_1 \in K_1'$ for some kind K_1'; likewise $\Gamma \vdash K_1 \in K_1'' \rightarrow^{?'} K_2'$ and $\Gamma \vdash T_1 \in K_1''$ for some kind K_1''. By Lemma C.1, $\Gamma \vdash S_1 \in K_1 \rightarrow K_2'$ and $\Gamma \vdash T_1 \in K_1 \rightarrow K_2'$, so the case follows with R-App$+_l$.

The other application cases are similar. □

Proof of Lemma 5.54 on page 81: By induction on the length of derivation of $\Gamma \vdash_{\mathcal{R}} S \leq T \in K$. Transitivity is solved by induction.

Case R-Refl:

$$\frac{S \longrightarrow^{*}_{\beta\top} U \qquad T \longrightarrow^{*}_{\beta\top} U \qquad \Gamma \vdash U \in K}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K}$$

From $\Gamma \vdash S \equiv S' \in K$ we get by Lemma 5.40 a type U_1 with $S' \longrightarrow^{*}_{\beta\top} U_1$ and $\Gamma \vdash U \equiv U_1 \in K$. Likewise there exists a type U_2 with $T' \longrightarrow^{*}_{\beta\top} U_2$ and $\Gamma \vdash U \equiv U_2 \in K$. By transitivity of equivalence (Lemma 5.34) $\Gamma \vdash U_1 \equiv U_2 \in K$, thus $\Gamma' \vdash_{\mathcal{R}} U_1 \leq U_2 \in K$ (Lemma 5.34 and Lemma 5.55), and the result follows by the expansion Lemma 5.48.

Case R-All:

$$\frac{\begin{array}{c} S \longrightarrow^{*}_{\beta\top} All(A{\leq}S_1{:}K_1)S_2 \qquad T \longrightarrow^{*}_{\beta\top} All(A{\leq}T_1{:}K_1)T_2 \\ \Gamma \vdash S_1 \equiv T_1 \in K_1 \qquad \Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{R}} S_2 \leq T_2 \in \star \end{array}}{\Gamma \vdash_{\mathcal{R}} S \leq T \in \star}$$

By Lemma 5.40 we get $S' \longrightarrow^{*}_{\beta\top} All(A{\leq}S_1'{:}K_1)S_2'$ and $\Gamma \vdash All(A{\leq}S_1{:}K_1)S_2 \equiv All(A{\leq}S_1'{:}K_1)S_2' \in \star$ for some type $All(A{\leq}S_1'{:}K_1)S_2'$. Likewise for T. Since by definition of equivalence on contexts, $\vdash \Gamma, A{\leq}S_1{:}K_1 \equiv \Gamma', A{\leq}S_1'{:}K_1$, again by induction $\Gamma', S_1'{:}K_1 \vdash_{\mathcal{R}} S_2' \leq T_2' \in \star$. The result follows by R-All and the expansion lemma.

Case R-Promote:

$$\frac{S \longrightarrow^{*}_{\beta\top} U \uparrow_{\Gamma} V \qquad \Gamma \vdash U \in K \qquad \Gamma \vdash_{\mathcal{R}} V \leq T \in K}{\Gamma \vdash_{\mathcal{R}} S \leq T \in K}$$

By Lemma 5.40 and 5.41, there exists a type V' with $S' \longrightarrow^{*}_{\beta\top} \uparrow_{\Gamma} V'$ and $\Gamma \vdash V \equiv V'' \in K$. Hence by induction $\Gamma' \vdash_{\mathcal{R}} V' \leq T' \in K$, and the case follows with R-Promote. □

Proof of Lemma 5.55 on page 82: By Lemma 5.29, $\Gamma \vdash S \in K$ and $\Gamma \vdash T \in K$. Proceed by straightforward induction on the structure of S and the generation lemma for kinds. □

Proof of Lemma 5.56 on page 82: By induction on the syntactic structure of S and T (since we use neither reduction, nor promotion, nor transitivity, which could destroy the syntactic form of the types involved, we can do so). So assume $\Gamma \vdash_{\mathcal{C}} S \leq T \in K$ and $\Gamma \vdash_{\mathcal{C}} T \leq S \in K$ by two cut-free derivations of this restricted form.

If one of the derivations ends with R-Refl, i.e., if $S = T$, the case is immediate by E-Refl. In the absence of promotion and reduction, the choice of rule for both derivations — besides reflexivity — is determined by the structure of S and T.

If $S = \mathit{Top}(K_S)$ and $T = \mathit{Top}(K_T)$, the case follows well-kindedness of subderivations — S and T are in normal form — and E-Top. The case for type variables is immediate, as R-Refl is the only rule applicable. The case of arrow-types and type operators by straightforward induction. We show the cases for All-types and one application case.

Case: $S = \mathit{All}(A{\leq}S_1{:}K_1)S_2$ and $\mathit{All}(A{\leq}T_1{:}K_1)T_2$
We are given:

$$\frac{\begin{array}{c}\Gamma \vdash S_1 \equiv T_1 \in K_1 \\ \Gamma,\, A{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \leq T_2 \in \star\end{array}}{\Gamma \vdash_{\mathcal{C}} \mathit{All}(A{\leq}S_1{:}K_1)S_2 \leq \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star} \quad \frac{\begin{array}{c}\Gamma \vdash T_1 \equiv S_1 \in K_1 \\ \Gamma,\, A{\leq}T_1{:}K_1 \vdash_{\mathcal{C}} T_2 \leq S_2 \in \star\end{array}}{\Gamma \vdash_{\mathcal{C}} \mathit{All}(A{\leq}T_1{:}K_1)T_2 \leq \mathit{All}(A{\leq}S_1{:}K_1)S_2 \in \star}$$

The derivation of neither $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \leq T_2 \in \star$ nor $\Gamma, A{\leq}T_1{:}K_1 \vdash_{\mathcal{C}} T_2 \leq S_2 \in \star$ contains an instance of R-Promote by assumption. By definition of equivalence of contexts $\vdash \Gamma, A{\leq}S_1{:}K_1 \equiv \Gamma, A{\leq}T_1{:}K_1$, so by Lemma 5.54, $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} T_2 \leq S_2 \in \star$. Moreover, S_2 and T_2 are well-kinded also in context $\Gamma, A{\leq}S_1{:}K_1$ (Lemma 5.32). By induction we thus get $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \equiv T_2 \in \star$ and hence with E-All $\Gamma \vdash_{\mathcal{C}} \mathit{All}(A{\leq}S_1{:}K_1)S_2 \equiv \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star$.

Case: $S = S_1\ S_2$ and $T = T_1\ T_2$
From the two derivations $\Gamma \vdash_{\mathcal{C}} S_1\ S_2 \gtrless T_1\ T_2 \in K_2$ without promotion we get independently from the special instance of the application rule that $\Gamma \vdash_{\mathcal{C}} S_1 \gtrless T_1 \in K_1 \rightarrow^{?} K$ for some kind K_1. Thus by induction $\Gamma \vdash_{\mathcal{C}} S_1 \equiv T_1 \in K_1 \rightarrow^{?} K$. By Corollary 5.35, the two equivalent types S_1 and T_1 bear the same minimal polarity as type operator. If, for instance, $\Gamma \vdash_{\mathcal{A}} S_1 +$ we get:

$$\frac{\begin{array}{c}\Gamma \vdash_{\mathcal{A}} S_1 + \\ \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \rightarrow K \\ \Gamma \vdash_{\mathcal{C}} S_2 \leq T_2 \in K_1\end{array}}{\Gamma \vdash_{\mathcal{C}} S_1\ S_2 \leq T_1\ T_2 \in K} \quad \frac{\begin{array}{c}\Gamma \vdash_{\mathcal{A}} S_1 + \\ \Gamma \vdash_{\mathcal{C}} T_1 \leq S_1 \in K_1 \rightarrow K \\ \Gamma \vdash_{\mathcal{C}} T_2 \leq S_2 \in K_1\end{array}}{\Gamma \vdash_{\mathcal{C}} T_1\ T_2 \leq S_1\ S_2 \in K}$$

and the result follows by induction and E-App+. The remaining cases are similar.
□

C.4 Subject reduction for subtyping

Proof of the substitution Lemma 5.57 on page 83: By induction on the syntactic structure of type S. Case 1 and 4 are already covered by Lemma 5.36, Lemma 5.39, and Lemma 5.55.

Case: $S = A'$
We distinguish, whether A and A' coincide. For $A = A'$ neither case 1 nor case 3 can hold. The type variable A is in normal form, so by well-kindedness of subderivations (Lemma 5.49) $\Gamma \vdash A \in K$, and thus by the generation Lemma 5.8 for kinds $\mathit{kind}_\Gamma A = K' \leq K$. Since we assumed at least one of the four cases to be true, by assumption — in case 4 using additionally Lemma 5.55 — $\Gamma_1 \vdash_\mathcal{C} T_1 \leq T_2 \in K'$. Hence by Lemma 5.52 $\Gamma_1 \vdash_\mathcal{C} T_1 \leq T_2 \in K$ for the weaker kind K. The result $\Gamma' \vdash_\mathcal{C} T_1 \leq T_2 \in K$ follows by the weakening lemma for subtyping (Lemma 5.50).

In case $A' \neq A$, the substitution has no effect and the result is immediate by reflexivity of subtyping, using the substitution property for kinds and contexts of Lemma 5.24.

Case: $S = \mathit{Top}(K'')$
Immediate with Lemma 5.24 and R-Top, as the substitution has no effect.

Case: $S = S_1 \to S_2$
In the first part we are given $\Gamma \vdash S_1 \to S_2 \circ_A$. Thus by the generation Lemma 5.8 $\Gamma \vdash S_1 \circ_A$ and $\Gamma \vdash S_2 \circ_A$. Again with the same generation lemma we get $\Gamma \vdash S_1 \in \star$ and $\Gamma \vdash S_2 \in \star$. By induction on part 1 of the lemma we can infer $\Gamma' \vdash_\mathcal{C} [T_2/A]S_1 \leq [T_1/A]S_1 \in \star$ and also $\Gamma' \vdash_\mathcal{C} [T_1/A]S_2 \leq [T_2/A]S_2 \in \star$, so that with R-Arrow we obtain $\Gamma \vdash_\mathcal{C} [T_1/A](S_1 \to S_2) \leq [T_2/A](S_1 \to S_2)$.

For part 2 we infer from $\Gamma \vdash S_1 \to S_2 +_A$ with the generation lemma $\Gamma \vdash S_1 +_A$ and $\Gamma \vdash S_2 -_A$. By induction on part 3 and 2 we get for the contravariant side of the arrow $\Gamma' \vdash_\mathcal{C} [T_2/A]S_1 \leq [T_1/A]S_1 \in \star$ and for the co-variant side $\Gamma' \vdash_\mathcal{C} [T_1/A]S_2 \leq [T_2/A]S_2 \in \star$. R-Arrow then gives:

$$\frac{\Gamma' \vdash_\mathcal{C} [T_2/A]S_1 \leq [T_1/A]S_1 \in \star \qquad \Gamma' \vdash_\mathcal{C} [T_1/A]S_2 \leq [T_2/A]S_2 \in \star}{\Gamma \vdash_\mathcal{C} [T_1/A](S_1 \to S_2) \leq [T_2/A](S_1 \to S_2) \in \star}$$

The third case is symmetric.

Case: $S = \mathit{All}(A'{\leq}S_1{:}K_1)S_2$
If one of the cases 1 – 3 holds, the statement $\Gamma \vdash \mathit{All}(A'{\leq}S_1{:}K_1)S_2\ ?_A$ implies $\Gamma, A'{\leq}S_1{:}K_1 \vdash S_2\ ?_A$ with $\Gamma \vdash S_1 \circ_A$ and $\Gamma, A'{\leq}S_1{:}K_1 \vdash S_2 \in \star$ (by the generation Lemma 5.8). Therefore $\vdash \Gamma, A'{\leq}S_1{:}K_1 \circ_A$ and the induction hypothesis applies, yielding $\Gamma', A'{\leq}[T_1/A]S_1{:}K_1 \vdash_\mathcal{C} [T_1/A]S_2 \leq [T_2/A]S_2 \in \star$. By the lemma for constant

substitution (Lemma 5.39) $\Gamma' \vdash_{\mathcal{C}} [T_1/A]S_1 \equiv [T_2/A]S_1 \in K_1$, so with R-ALL we conclude:

$$\frac{\Gamma' \vdash_{\mathcal{C}} [T_1/A]S_1 \equiv [T_2/A]S_1 \in K_1 \qquad \Gamma', A'{\leq}[T_1/A]S_1{:}K_1 \vdash_{\mathcal{C}} [T_1/A]S_2 \leq [T_2/A]S_2 \in \star}{\Gamma' \vdash_{\mathcal{C}} [T_1/A](All(A'{\leq}S_1{:}K_1)S_2) \leq [T_2/A](All(A'{\leq}S_1{:}K_1)S_2) \in \star}$$

If case 4 holds, we get by generation $\Gamma, A'{\leq}S_1{:}K_1 \vdash S_2 \pm_A$ and the result follows again by induction.

Case: $S = Fun(A'{:}K'')S'$
Similar, using the generation lemma and by induction.

Case: $S = S_1\ S_2$
We are given $\Gamma \vdash S_1\ S_2 \in K$. Again by the generation lemma $\Gamma \vdash K_1 \to K$ and $\Gamma \vdash S_2 \in K_1$ for some kind K_1.

We have to distinguish according to the polarity of the variables for the respective parts of the lemma. In all cases the following implication holds with the generation Lemma 5.8: if $\Gamma \vdash S_1\ S_2\ ?_A$, then also $\Gamma \vdash S_1\ ?_A$. By the generation Lemma 5.8, the subterm S_1 is well-kinded in Γ; thus by induction $\Gamma' \vdash_{\mathcal{C}} [T_1/A]S_1 \leq [T_2/A]S_1 \in K_1 \to^{?} K$.

For the argument S_2, we use the generation Lemma 5.21 for minimal kinds to infer in each case from the minimal polarity of S_1 as type operator the polarity of A in S_2.

Subcase: $\Gamma \vdash S_1\ S_2\ +_A$
for the second part of the lemma. We have further to distinguish according to the minimal polarity of S_1 as type operator. If, for instance, $\Gamma \vdash_{\mathcal{A}} S_1 \pm$ is minimal, then $\Gamma \vdash S_2\ \circ_A$, so we the result follows by induction, preservation of kinding under substitution, and R-APP$+_l$. The remaining cases are similar. □

Proof of the substitution Lemma 5.58 on page 83: Inducing on the length of derivation, we treat all cases of the lemma at the same time.

Case R-REFL:
$$\frac{S \longrightarrow^{*}_{\beta\top} U \qquad T \longrightarrow^{*}_{\beta\top} U \qquad \Gamma \vdash U \in K}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K}$$

In the first part we are given $\Gamma \vdash S\ \circ_A$. By subject reduction (Lemma 5.25) U is well-kinded with $\Gamma \vdash U\ \circ_A$. Using the substitution Lemma 5.57 $\Gamma' \vdash_{\mathcal{C}} [V_1/A]U \leq [V_2/A]U \in K$, so the result follows by expansion lemma.

The other parts are solved analogously by the respective parts of Lemma 5.57. The cases for $\Gamma \vdash T\ ?_A$ on the right-hand side are identical.

Case R-PROMOTE: $$\frac{S \longrightarrow^{*}_{\beta\top} U \uparrow_{\Gamma} S' \qquad \Gamma \vdash U \in K \qquad \Gamma \vdash_{\mathcal{C}} S' \leq T \in K}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K}$$

We are given $\Gamma \vdash S\ ?_A$ with the polarity of A according to the four parts of the lemma. By definition of promotion, we know $S \longrightarrow^{*}_{\beta\top} A'\ S_1 \ldots S_n \uparrow_{\Gamma} \Gamma(A')\ S_1 \ldots S_n = S'$. We distinguish the following two subcases:

Subcase: $A \neq A'$

By preservation of polarity under reduction (Lemma 5.25) we get $\Gamma \vdash U\ ?_A$. In part 1 – 3 of the lemma we further get by the preservation of polarity under promotion (Lemma 5.47) $\Gamma' \vdash S'\ ?_A$. The side condition of Lemma 5.47 is satisfied by the form of the context $\Gamma = \Gamma_1,\ A{:}K',\ \Gamma_2$ with $\vdash \Gamma \circ_A$. By induction we get $\Gamma' \vdash_{\mathcal{C}} [V_1/A](\Gamma(A')S_1 \ldots S_n) \leq [V_2/A]T \in K$. Now,

$$\begin{aligned}[V_1/A](\Gamma(A')\ S_1 \ldots S_n) &= [V_1/A](\Gamma(A'))\ [V_1/A]S_1 \ldots [V_1/A]S_n \\ &= \Gamma'(A')\ [V_1/A]U_1 \ldots [V_1/A]U_n,\end{aligned}$$

and by Lemma A.3, $[V_1/A]S \longrightarrow^{*}_{\beta\top} A'\ [V_1/A]S_1 \ldots [V_1/A]S_n$. So we can use the rule of promotion to conclude (the kinding premise follows by subject reduction for kinding and preservation of kinding under substitution):

$$\frac{\begin{array}{c}[V_1/A]S \longrightarrow^{*}_{\beta\top} \uparrow_{\Gamma'} \Gamma'(A')\ [V_1/A]S_1 \ldots [V_1/A]S_n \\ \Gamma \vdash A'\ [V_1/A]S_1 \ldots [V_1/A]S_n \in K \quad \Gamma' \vdash_{\mathcal{C}} \Gamma'(A')\ [V_1/A]S_1 \ldots [V_1/A]S_n \leq [V_2/A]T \in K\end{array}}{\Gamma' \vdash_{\mathcal{C}} [V_1/A]S \leq [V_2/A]T \in K}$$

Part 4 is simpler.

The cases for T instead of S are solved straightforwardly by induction.

Subcase: $A = A'$

In this case $S' = \Gamma(A)\ S_1 \ldots S_n = \mathit{Top}(K')\ S_1 \ldots S_n$ (Remember that $\Gamma_1,\ A{:}K',\ \Gamma_2$ abbreviates $\Gamma_1,\ A{\leq}\mathit{Top}(K'){:}K',\ \Gamma_2$). By well-kindedness of subderivations and maximality of *Top* (Lemma 5.51) there exists a kind K' with $T \longrightarrow^{*}_{\beta\top} \mathit{Top}(K'')$ and $\Gamma \vdash \mathit{Top}(K'') \in K$, so the case follows directly with R-TOP.

Case R-TOP: $$\frac{\begin{array}{cc}S \longrightarrow^{*}_{\beta\top} S' & T \longrightarrow^{*}_{\beta\top} \mathit{Top}(K''') \\ \Gamma \vdash S' \in K & \Gamma \vdash_{\mathcal{C}} \mathit{Top}(K''') \in K\end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K}$$

By preservation of kinding under substitution, Lemma A.3, and R-TOP.

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

In the first part we obtain by subject reduction for kinds (Lemma 5.25) $\Gamma \vdash S_1 \to$

$S_2 \circ_A$. By the generation Lemma 5.8 further $\Gamma \vdash S_1 \circ_A$ and $\Gamma \vdash S_2 \circ_A$. The case follows by induction, rule R-Arrow and Lemma A.3.

The other cases are solved similarly.

Case R-All:

$$\frac{S \longrightarrow^*_{\beta\top} All(A'{\leq}S_1{:}K_1)S_2 \qquad T \longrightarrow^*_{\beta\top} All(A'{\leq}T_1{:}K_1)T_2 \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \qquad \Gamma,\, A'{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \leq T_2 \in \star}{\Gamma \vdash_{\mathcal{C}} S \leq T \in \star}$$

We are given $\Gamma \vdash S\ ?_A$ with the polarity of A according to the four parts of the lemma. By preservation of polarity under reduction (Lemma 5.25) and the generation lemma $\Gamma,\, A'{\leq}S_1{:}K_1 \vdash S_2\ ?_A$. Also by generation we get in part 1 – 3 $\Gamma \vdash S_1 \circ_A$. By Lemma 5.39 we obtain $\Gamma' \vdash [V_1/A]S_1 \equiv [V_2/A]T_1 \in K_1$ and we can conclude with R-All:

$$\frac{\Gamma' \vdash [V_1/A]S_1 \equiv [V_2/A]T_1 \in K_1 \qquad \Gamma',\, A'{\leq}[V_1/A]S_1{:}K_1 \vdash_{\mathcal{C}} [V_1/A]S_2 \leq [V_2/A]T_2 \in \star}{\Gamma' \vdash_{\mathcal{C}} [V_1/A](All(A'{\leq}S_1{:}K_1)S_2 \leq [V_2/A](All(A'{\leq}T_1{:}K_1)T_2 \in \star}$$

So the case follows with the expansion lemma. Part 4 of the lemma is similar.

Case R-Abs:

Similarly.

Case R-App$+_l$:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{A}} S_1 + \qquad \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \rightarrow K_2 \qquad \Gamma \vdash_{\mathcal{C}} S_2 \leq T_2 \in K_1}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_2}$$

We are given $\Gamma \vdash S\ ?_A$ according to the part of the lemma; by preservation of polarity under reduction $\Gamma \vdash S_1\ S_2\ ?_A$, by the generation Lemma 5.8, and by induction $\Gamma' \vdash_{\mathcal{C}} [V_1/A]S_1 \leq [V_2/A]T_1 \in K_1 \rightarrow K_2$.

Since the system always chooses the application rule according to the minimal polarity of one of the two applicators, we can use Lemma 5.21 to gain information about the polarity of A in S_2. By the choice of rule, the minimal polarity of S_1 here is $\Gamma \vdash_{\mathcal{A}} S_1 +$.

In the first part of the lemma we get from $\Gamma \vdash S_1\ S_2 \circ_A$ by the generation Lemma 5.21 for minimal kinds $\Gamma \vdash S_2 \circ_A$. Hence by induction on the first part $\Gamma' \vdash_{\mathcal{C}} [V_1/A]S_2 \leq [V_2/A]T_2 \in K_1$. So by preservation of kinding under substitution we get as kinding statement $\Gamma' \vdash [V_1/A]S_1 +$. Note that we cannot guarantee $\Gamma' \vdash_{\mathcal{A}} [V_1/A]S_1 +$, i.e. we cannot guarantee that after substitution the minimal polarity of $[V_1/A]S_1$ as type operator is not better than monotone (i.e. constant) since we have made no assumption about the minimal polarity of V_1 in Γ_1, as required for preservation of minimal kinds in Lemma 5.24. Nevertheless, the minimal polarity of $[V_1/A]S_1$ must be at least monotone. If it is, by Lemma A.3 and R-App$+_l$ we conclude:

$$\frac{[V_1/A]S \longrightarrow^*_{\beta\top} [V_1/A](S_1\ S_2) \qquad [V_2/A]T \longrightarrow^*_{\beta\top} [V_2/A](T_1\ T_2) \qquad \Gamma' \vdash_{\mathcal{A}} [V_1/A]S_1\ + \qquad \Gamma' \vdash_{\mathcal{C}} [V_1/A]S_1 \leq [V_2/A]T_1 \in K_1 \to K_2 \qquad \Gamma' \vdash_{\mathcal{C}} [V_1/A]S_2 \leq [V_2/A]T_2 \in K_1}{\Gamma' \vdash_{\mathcal{C}} [V_1/A]S \leq [V_2/A]T \in K_2}$$

If the substitution happens to result in $[V_1/A]S_1$ or $[V_2/A]T_1$ being a constant operator, the case follows by one of the constant application rules.

In part 3 we can use the same lemma to get from $\Gamma \vdash_{\mathcal{A}} S_1 +$ as minimal polarity that $\Gamma \vdash S_2 -_A$. Hence by induction on part 3 $\Gamma' \vdash_{\mathcal{C}} [V_2/A]S_2 \leq [V_1/A]T_2 \in K_1$ and the case follows by as in the case above by R-App$+_l$ or one of the constant application cases.

Part 2 is similar.

Case R-App$+_r$:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{A}} T_1\ + \qquad \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash_{\mathcal{C}} S_2 \leq T_2 \in K_1}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_2}$$

Similar, but not symmetric, to the previous case. Since now T_1 determines the choice of rule, we have to check some more cases for the polarity of A in S.

For S_1 we can argue in the same way as in the case of R-App$+_l$ to infer $\Gamma' \vdash_{\mathcal{C}} [V_1/A]S_1 \leq [V_2/A]S_2 \in K_1 \to K_2$.

Here we do not know the minimal polarity of S_1, we only know that it cannot be better than monotone. By Lemma 5.21 we can again check in all parts, that A occurs appropriately in S_2. If, e.g. in the second part only $\Gamma \vdash S \pm_A$ as minimal polarity, we get $\Gamma \vdash S_2 \circ_A$. So we can use induction in the first part to obtain $\Gamma' \vdash [V_1/A]S_1 \leq [V_2/A]T_1 \in K$. By preservation of kinding under substitution, Lemma A.3, and R-App$+_r$ or one of the constant application rules we can conclude also this case.

Case R-App$\pm$:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_2}$$

As in the cases for monotone polarity of A, we get in all four parts by preservation of polarity under reduction, the generation lemma, and induction $\Gamma' \vdash_{\mathcal{C}} [V_1/A]S_1 \leq [V_2/A]T_1 \in K_1 \to K_2$.

We know that the minimal polarity of S_1 as type operator cannot be better than $\pm$. Hence by Lemma 5.21 we get in parts 1 – 3 of the lemma $\Gamma \vdash S_1 \circ_A$, and in part 4 $\Gamma \vdash S_2 \pm_A$.

By preservation of equivalence under constant substitution (Lemma 5.39) we get $\Gamma' \vdash [V_1/A]S_2 \equiv [V_2/A]T_2 \in K_1$. For part 4 we obtain this statement from the assumption $\Gamma_1 \vdash V_1 \equiv V_2 \in K'$ using Lemma 5.37. Thus we conclude the case by R-APP$\pm$ or, if the substitution results in a better polarity, by the corresponding application rule, with the help of Lemma 5.55 in the monotone or antimonotone case.

The cases for T instead of S are treated similarly. The interesting case is the one for universally quantified types.

Case R-ALL:

$$\frac{\begin{array}{c} S \longrightarrow^{*}_{\beta\top} \mathit{All}(A'{\leq}S_1{:}K_1)S_2 \qquad T \longrightarrow^{*}_{\beta\top} \mathit{All}(A'{\leq}T_1{:}K_1)T_2 \\ \Gamma \vdash S_1 \equiv T_1 \in K_1 \qquad \Gamma,\, A'{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \leq T_2 \in \star \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T \in \star}$$

The case is treated analogously to the one where the type, whose polarity in A determined the case, was on the left-hand side.

The only part missing here is to derive that (in the cases 1 – 3) $\Gamma \vdash S_1 \circ_A$. This follows from $\Gamma \vdash T_1 \circ_A$ and Lemma 5.32.[2] □

Proof of Lemma 5.59 on page 84: Part 1 by induction on the derivation of $\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K_1')T \in K_1 \to K_2$.

Case R-REFL:

$$\frac{\begin{array}{c} S \longrightarrow^{*}_{\beta\top} \mathit{Fun}(A{:}K_1')T' \qquad \mathit{Fun}(A{:}K_1')T \longrightarrow^{*}_{\beta\top} \mathit{Fun}(A{:}K_1')T' \\ \Gamma \vdash \mathit{Fun}(A{:}K_1')T' \in K_1 \to K_2 \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K_1')T \in K_1 \to K_2}$$

We know that

$$\begin{array}{l} S\ U_1 \longrightarrow^{*}_{\beta\top} (\mathit{Fun}(A{:}K_1')T')U_1 \longrightarrow_{\beta} [U_1/A]T' \quad \text{and} \\ T\ U_2 \longrightarrow^{*}_{\beta\top} (\mathit{Fun}(A{:}K_1')T')U_2 \longrightarrow_{\beta} [U_2/A]T'. \end{array}$$

By preservation of kinding under reduction and generation for kinds we get $\Gamma \vdash U_1,\, U_2 \in K_1'$. By well-kindedness of subderivations and the generation lemma for kinds $\Gamma',\, A{:}K_1' \vdash T'^{!} \in K_2$. Thus by the substitution Lemma 5.57 $\Gamma \vdash_{\mathcal{C}} [U_1/A]T'^{!} \leq [U_2/A]T'^{!} \in K_2$. By Lemma A.3 $[U_2/A]T \longrightarrow^{*}_{\beta\top} [U_2/A]T'$ and the case follows by the expansion lemma.

Case R-PROMOTE:

$$\frac{\begin{array}{c} S \longrightarrow^{*}_{\beta\top} U \uparrow_{\Gamma} S' \qquad \Gamma \vdash U \in K_1 \to K_2 \\ \Gamma \vdash_{\mathcal{C}} S' \leq \mathit{Fun}(A{:}K_1')T \in K_1 \to K_2 \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K_1')T \in K_1 \to K_2}$$

By definition of promotion $S \longrightarrow^{*}_{\beta\top} A'\ V_1 \ldots V_n \uparrow_{\Gamma} \Gamma(A')\ V_1 \ldots V_n = S'$. By preservation of kinding under reduction and promotion, the induction hypothesis applies,

[2]Note that at this place we can rely on the stricter relation $\Gamma \vdash S_1 \equiv T_1 \in K_1$, as opposed to $\Gamma \vdash S_1 \gtrless T_1 \in K$. Unlike the latter, the equivalence relation allows to infer from $\Gamma \vdash T \circ_A$ that also $\Gamma \vdash S_1 \circ_A$. Thus by the stricter relation of equivalence we can deal deal with the asymmetric case of All-types here, which was the main motivation for using equivalence.

yielding $\Gamma \vdash_{\mathcal{C}} \Gamma(A')\ V_1 \ldots V_n\ U_1 \leq [U_2/A]T' \in K_2$, and the result follows with R-Promote.

Case R-Abs:

$$\frac{S \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')S' \qquad \mathit{Fun}(A{:}K_1')T \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')T' \qquad K_1 \leq K_1' \qquad \Gamma, A{:}K_1' \vdash_{\mathcal{C}} S' \leq T' \in K_2}{\Gamma \vdash_{\mathcal{C}} S \leq \mathit{Fun}(A{:}K_1')T \in K_1 \to K_2}$$

The assumption $\Gamma \vdash \mathit{Fun}(A{:}K_1')T \circ$ implies $\Gamma, A{:}K_1' \vdash T' \circ_A$ (using preservation of polarity under reduction and the generation Lemma 5.8). By preservation of kinding under reduction and the generation lemma for kinds $\Gamma \vdash U_1 \in K_1'$ and $\Gamma \vdash U_2 \in K_1'$, hence by the substitution Lemma 5.58 $\Gamma \vdash_{\mathcal{C}} [U_1/A]S' \leq [U_2/A]T' \in K_2$, and with the expansion lemma $\Gamma \vdash_{\mathcal{C}} S\ U_1 \leq [U_2/A]T \in K_2$. The remaining cases are solved similarly.

In the second part of the lemma, R-Refl and R-Abs are solved analogously the corresponding cases in the first part.

Case R-Top:

$$\frac{T \longrightarrow^*_{\beta\top} \mathit{Top}(K') \qquad \Gamma \vdash \mathit{Fun}(A{:}K_1')S \in K_1 \to K_2 \qquad \Gamma \vdash \mathit{Top}(K') \in K_1 \to K_2}{\Gamma \vdash_{\mathcal{C}} \mathit{Fun}(A{:}K_1')S \leq T \in K_1 \to K_2}$$

Since $T\ U_2$ is well-kinded, we get $T\ U_2 \longrightarrow^*_{\beta\top} \mathit{Top}(K'')$ with $\Gamma \vdash \mathit{Top}(K'') \in K_2$, and the case follows by R-Top. □

Proof of Lemma 5.60 on page 84: By induction on the depth of inference, similar to the proof of Lemma 5.59. □

Proof of Lemma 5.61 on page 84: The part for the reduction step on the right is proven by induction over the derivation of $\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K_1')T)\ U \in K$.

Most cases are straightforward. With the exception of the rules for application, we get by Corollary A.5 that $(\mathit{Fun}(A{:}K_1')T)\ U \longrightarrow^*_{\beta\top} W$ implies $[U/A]T \longrightarrow^*_{\beta\top} W$. Hence in order to derive $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T \in K'$, one can directly use the derivation of the original statement $\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K_1')T)\ U \in K'$.

The cases for application are more difficult, since by contracting the outermost redex, the use of an application rule, that might have led in a last derivation step to the statement $\Gamma \vdash_{\mathcal{C}} S \leq [U/A]T \in K$, can be rendered impossible.

Case R-App$+_l$:

$$\frac{S \longrightarrow^*_{\beta\top} V_1\ V_2 \qquad (\mathit{Fun}(A{:}K_1')T)\ U \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{A}} V_1 + \qquad \Gamma \vdash_{\mathcal{C}} V_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash_{\mathcal{C}} V_2 \leq T_2 \in K_1}{\Gamma \vdash_{\mathcal{C}} S \leq (\mathit{Fun}(A{:}K_1')T)\ U \in K_2}$$

We distinguish, whether the outermost redex $(\mathit{Fun}(A{:}K_1')T)\ U$ gets contracted in the reduction sequence or not. If it is, the case is easy and similar to the ones for the other rules:

Subcase: $(\mathit{Fun}(A{:}K_1')T)\ U \longrightarrow^*_{\beta\top} (\mathit{Fun}(A{:}K_1')T')\ U' \longrightarrow_{\beta\top} [U'/A]T' \longrightarrow^*_{\beta\top} T_1\ T_2$
By Lemma A.2 also $[U/A]T \longrightarrow^*_{\beta\top} [U'/A]T' \longrightarrow^*_{\beta\top} T_1\ T_2$, the case is immediate by rule R-App+$_l$ and preservation of kinding under reduction.

Subcase: $(\mathit{Fun}(A{:}K_1')T)\ U \longrightarrow^*_{\beta\top} (\mathit{Fun}(A{:}K_1')T_1')\ T_2$
where $\mathit{Fun}(A{:}K_1')T \longrightarrow^*_{\beta\top} \mathit{Fun}(A{:}K_1')T_1' = T_1$ and $U \longrightarrow^*_{\beta\top} T_2$. So we are given $\Gamma \vdash_{\mathcal{C}} V_1 \leq \mathit{Fun}(A{:}K_1')T_1' \in K_1 \rightarrow K_2$. By well-kindedness of subderivations this statement implies $\Gamma \vdash \mathit{Fun}(A{:}K_1')T_1'^{!} \in K_1 \rightarrow K_2$ and thus by the generation lemma for kinds $K_1 \leq K_1'$. Again using well-kindedness of subderivations and Lemma 5.52, the statement $\Gamma \vdash_{\mathcal{C}} V_2 \leq T_2 \in K_1$ implies that also $\Gamma \vdash_{\mathcal{C}} V_2 \leq T_2 \in K_1'$ for the weaker kind K_1'. So by part of 1 Lemma 5.59 $\Gamma \vdash_{\mathcal{C}} V_1\ V_2 \leq [T_2/A]T_1' \in K_1$. Since $[U/A]T \longrightarrow^*_{\beta\top} [T_2/A]T_1'$ (Lemma A.2), the case follows with the expansion lemma.

The remaining application cases are are similar.

The second part of the lemma, dealing with an outer reduction step on the left-hand side, is shown by induction on the derivation of $\Gamma \vdash_{\mathcal{C}} (\mathit{Fun}(A{:}K_1')S)\ U \leq T \in K$.

The cases for R-Refl, R-Arrow, R-All, and R-Abs are symmetric to the corresponding cases in the first part of the Lemma; R-Top and R-Promote are solved by simple induction. The cases for application are solved analogously to the respective ones in the first part, using part 2 of Lemma 5.59. □

Proof of parallel reduction Lemma 5.62 on page 84: Both parts by induction on the length of derivation. The properties of the parallel reduction relation (Lemma A.6) allow to use induction in all cases, especially in the case of R-Promote.

Case R-Refl:

$$\frac{S \longrightarrow^*_{\beta\top} U \qquad T \longrightarrow^*_{\beta\top} U \qquad \Gamma \vdash U \in K}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K}$$

By Lemma A.6 $U \twoheadrightarrow_{\beta\top} U'$, $S' \longrightarrow^*_{\beta\top} U'$ and $T \longrightarrow^*_{\beta\top} U'$ for some type U'. Thus the case follows by R-Refl and preservation of kinding under reduction.

Case R-Top:

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

By subject reduction for kinding (Lemma 5.25), $\Gamma' \vdash S' \in K$ and $\Gamma' \vdash \mathit{Top}(K') \in K$, and the case follows with R-Top.

Case R-Promote:

$$\frac{S \longrightarrow^*_{\beta\top} W \uparrow_\Gamma U \qquad \Gamma \vdash W \in K \qquad \Gamma \vdash_{\mathcal{C}} U \leq T \in K}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K}$$

By the definition of promotion $W = A\ S_1 \ldots S_n \uparrow_\Gamma \Gamma(A)\ S_1 \ldots S_n = U$. By Lemma A.6 there are types $S_1', \ldots, S_n'$ with $S_i \twoheadrightarrow_{\beta\top} S_i'$ such that:

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

The case follows by induction, the fact that $\Gamma(A)\ S_1 \dots S_n \twoheadrightarrow_{\beta\top} \Gamma'(A)\ S'_1 \dots S'_n$, and R-PROMOTE.

Case R-ARROW:

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

By Lemma A.6 and the definition of parallel reduction there are two types S'_1 and S'_2 with $S_1 \twoheadrightarrow_{\beta\top} S'_1$ and $S_2 \twoheadrightarrow_{\beta\top} S'_2$ such that:

$$\begin{tikzcd} S \arrow[r, two heads, "\beta\top"'] \arrow[d, "*"', "\beta\top"] & S' \arrow[d, "*"', "\beta\top"] \\ S_1 \to S_2 \arrow[r, two heads, "\beta\top"'] & S'_1 \to S'_2. \end{tikzcd}$$

By well-kindedness of subderivations (Lemma 5.49) the types S_i and T_i are well-kinded in context Γ. So by induction $\Gamma' \vdash_{\mathcal{C}} T_1 \leq S'_1 \in \star$ and $\Gamma' \vdash_{\mathcal{C}} S'_2 \leq T_2 \in \star$. By R-ARROW we obtain $\Gamma \vdash_{\mathcal{C}} S' \leq T_1 \to T \in \star$.

Case R-ALL:

$$\frac{\begin{array}{cc} S \longrightarrow^*_{\beta\top} All(A{\leq}S_1{:}K_1)S_2 & T \longrightarrow^*_{\beta\top} All(A{\leq}T_1{:}K_1)T_2 \\ \Gamma \vdash S_1 \equiv T_1 \in K_1 & \Gamma,\, A{\leq}S_1{:}K_1 \vdash_{\mathcal{C}} S_2 \leq T_2 \in \star \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T \in \star}$$

By Lemma A.6 and the definition of parallel reduction there are two types S'_1 and S'_2 such that:

$$\begin{tikzcd} S \arrow[r, two heads, "\beta\top"'] \arrow[d, "*"', "\beta\top"] & S' \arrow[d, "*"', "\beta\top"] \\ All(A{\leq}S_1{:}K_1)S_2 \arrow[r, two heads, "\beta\top"'] & All(A{\leq}S'_1{:}K_1)S'_2. \end{tikzcd}$$

By induction $\Gamma', A{\leq}S_1'{:}K_1 \vdash_{\mathcal{C}} S_2' \leq T_2 \in \star$. From $\Gamma \vdash S_1 \equiv T_1 \in K_1$ and $S_1 \twoheadrightarrow_{\beta\top} S_1'$, we get by Lemma 5.40 some type T_1' with

$$\begin{tikzcd}
S_1 \arrow[r, equal, "\Gamma", "K_1"'] \arrow[d, "\beta\top"', "*"] & T_1 \arrow[d, "\beta\top"', "*"] \\
S_1' \arrow[r, equal, "\Gamma", "K_1"'] & T_1',
\end{tikzcd}$$

so by Lemma 5.32 $\Gamma' \vdash S_1' \equiv T_1' \in K_1$, and by R-All:

$$\frac{\begin{array}{cc} S' \longrightarrow^*_{\beta\top} All(A{\leq}S_1'{:}K_1)S_2' & T \longrightarrow^*_{\beta\top} All(A{\leq}T_1'{:}K_1)T_2' \\ \Gamma' \vdash S_1' \equiv T_1' \in K_1 & \Gamma', A{\leq}S_1'{:}K_1 \vdash_{\mathcal{C}} S_2' \leq T_2' \in \star \end{array}}{\Gamma' \vdash_{\mathcal{C}} S' \leq T \in \star}$$

Case R-Abs:
Analogous.

Case R-App$+_l$:

$$\frac{\begin{array}{ccc} & S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 & \\ \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \to K_2 & \Gamma \vdash_{\mathcal{A}} S_1 + & \Gamma \vdash_{\mathcal{C}} S_2 \leq T_2 \in K_1 \end{array}}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_2}$$

By Lemma A.6 there exists a type R such that:

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

By well-kindedness of subderivations S_1 and T_1 are well-kinded in Γ. We have to distinguish according to the form of S_1; the interesting case is, where it is a type operator.

Subcase: $S_1 = A$ or $S_1 = V_1\ V_2$
By definition of parallel reduction, $R = S_1'\ S_2'$ with $S_1 \twoheadrightarrow_{\beta\top} S_1'$ and $S_2 \twoheadrightarrow_{\beta\top} S_2'$. By preservation of kinding under reduction, Lemma 5.32, and reflexivity for equivalence $\Gamma' \vdash_{\mathcal{A}} S_1' +$ or $\Gamma' \vdash_{\mathcal{A}} S_1' \circ$. In the monotone case we can conclude by induction and R-App$+_l$:

$$\frac{\begin{array}{c} S' \longrightarrow^*_{\beta\top} S'_1\ S'_1 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \\ \Gamma' \vdash_{\mathcal{C}} S'_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma' \vdash_{\mathcal{A}} S'_1 + \qquad \Gamma' \vdash_{\mathcal{C}} S'_2 \leq T_2 \in K_1 \end{array}}{\Gamma' \vdash_{\mathcal{C}} S' \leq T \in K_2}$$

If S'_1 is a constant operators we can finish with R-APP$\circ_l$ instead.

Subcase: $S_1 = Top(K')$
This case cannot occur a subcase for monotone application, since maximal types are constant operators.

Subcase: $S_1 = Fun(A{:}K'_1)S'_1$
We are given $(Fun(A{:}K'_1)S_1)\ S_2 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} R$. By the definition of parallel reduction we can distinguish the following two subcases: one, where the outer redex does not get contracted in the parallel step $(Fun(A{:}K'_1)S'_1)\ S_2 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} R$, the second one, where the outer redex disappears.

Subsubcase: $R = (Fun(A{:}K'_1)S''_1)\ S_2$
$S_1 = Fun(A{:}K'_1)S'_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} Fun(A{:}K'_1)S''_1$
$S_2 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} S'_2$
This easier subcase is solved by induction and rule R-APP$+_l$, using subject reduction for kinding and the generation lemma (once again the case where $\Gamma' \vdash_{\mathcal{A}} Fun(A{:}K'_1)S''_1\ \circ$ is easier):

$$\frac{\begin{array}{c} S' \longrightarrow^*_{\beta\top} (Fun(A{:}K'_1)S''_1)\ S'_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \\ \Gamma' \vdash_{\mathcal{C}} Fun(A{:}K'_1)S''_1 \leq T_1 \in K_1 \to K_2 \quad \Gamma' \vdash_{\mathcal{A}} Fun(A{:}K'_1)S''_1 + \quad \Gamma' \vdash_{\mathcal{C}} S'_1 \leq T_2 \in K_1 \end{array}}{\Gamma' \vdash_{\mathcal{C}} S' \leq T \in K_2}$$

Subsubcase: $R = [S'_2/A]S''_1$
$S'_1 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} S''_1$
$S_2 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} S'_2$
We are given $\Gamma \vdash_{\mathcal{C}} Fun(A{:}K'_1)S'_1 \leq T_1 \in K_1 \to K_2$. By induction we get $\Gamma' \vdash_{\mathcal{C}} Fun(A{:}K'_1)S''_1 \leq T_1 \in K_1 \to K_2$. Hence by R-APP$+_l$:

$$\frac{\Gamma' \vdash_{\mathcal{C}} Fun(A{:}K'_1)S''_1 \leq T_1 \in K_1 \to K_2 \quad \Gamma' \vdash_{\mathcal{A}} Fun(A{:}K'_1)S''_1 + \quad \Gamma' \vdash_{\mathcal{C}} S'_2 \leq T_2 \in K_1}{\Gamma' \vdash_{\mathcal{C}} (Fun(A{:}K'_1)S''_1)\ S'_2 \leq T_1\ T_2 \in K_2}$$

Further by Lemma 5.61, we can perform one outer reduction step on the left-hand side, yielding $\Gamma' \vdash_{\mathcal{C}} [S'_2/A]S''_1 \leq T_1\ T_2 \in K_2$. As $S' \longrightarrow^*_{\beta\top} (Fun(A{:}K'_1)S''_1)\ S'_2 \longrightarrow_\beta [S'_2/A]S''_1$ and $T \longrightarrow^*_{\beta\top} T_1\ T_2$ we conclude with the expansion lemma $\Gamma' \vdash_{\mathcal{C}} S' \leq T \in K_2$.

Case R-APP∘:

$$\frac{S \longrightarrow^*_{\beta\top} S_1\ S_2 \qquad T \longrightarrow^*_{\beta\top} T_1\ T_2 \qquad \Gamma \vdash_{\mathcal{C}} S_1 \leq T_1 \in K_1 \to K_2 \qquad \Gamma \vdash S_1 \circ}{\Gamma \vdash_{\mathcal{C}} S \leq T \in K_2}$$

Similar. The only difference is, that now additionally the following case is possible: $S \longrightarrow^*_{\beta\top} S_1\ S_2 = \mathit{Top}(K')\ S_2$. By preservation of kinding under reduction and the generation lemma for kinds $K' = K'_1 \to^{?'} K'_2$. By Lemma A.6 there exists a type R such that:

$$\begin{tikzcd}
S \arrow[r, twoheadrightarrow, "\beta\top"'] \arrow[d, "*"', "\beta\top"] & S' \arrow[d, "*"', "\beta\top"] \\
\mathit{Top}(K'_1 \to^{?'} K'_2)\ S_2 \arrow[r, twoheadrightarrow, "\beta\top"'] & R
\end{tikzcd}$$

We distinguish whether the parallel step from $\mathit{Top}(K'_1 \to^{?'} K'_2)\ S_2$ to R contracts the $\top$-redex or not. If $\mathit{Top}(K')\ S_2 \longrightarrow\!\!\!\!\!\rightarrow_{\beta\top} \mathit{Top}(K'_1 \to^{?'} K'_2)\ S'$, the case follows by induction. If $\mathit{Top}(K'_1 \to^{?'} K'_2)\ S_2 \longrightarrow_{\top} \mathit{Top}(K'_2)$, the case follows by Lemma 5.60.

The other cases are similar.

The second part of the lemma, dealing with a parallel reduction step on the right-hand side, is analogous. The critical cases are again handled by Lemma 5.61. The case for R-PROMOTE is easier. □

Proof of Corollary 5.63 on page 85: If we can do one $\longrightarrow\!\!\!\!\!\rightarrow_{\beta\top}$-step we can do many. By Fact A.2, $\longrightarrow\!\!\!\!\!\rightarrow^*_{\beta\top}$ equals $\longrightarrow^*_{\beta\top}$. □

Proof of Lemma 5.64 on page 85: By Corollary 5.63 and Lemma 5.48. □

C.5 Strong normalization

Proof of Lemma 5.66 on page 86: By induction on the length of derivation of $\Gamma_1 \vdash T \in K$. We show the case for universally quantified types.

Case K-ALL:

$$\frac{\Gamma_1,\ A{\leq}T_1{:}K_1 \vdash T_2 \in \star}{\Gamma_1 \vdash \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

We distinguish where the reduction step takes place.

Subcase: $All(A{\leq}T_1{:}K_1)T_2 \longrightarrow_{\Gamma_1} All(A{\leq}T_1{:}K_1)T_2'$
where $T_2 \longrightarrow_{\Gamma_1,\, A\leq T_1:K_1} T_2'$. We know that $\vdash \Gamma_1, A{\leq}T_1{:}K_1 \equiv \Gamma_2, A{\leq}T_1{:}K_1$. Hence by induction

$$\begin{tikzcd}
T_2' \arrow[r, dotted, no head, "\Gamma_1,\, A\leq T_1:K_1", "\star"'] & T_2'' \\
T_2, \arrow[u, "\Gamma_1,\, A\leq T_1:K_1"] \arrow[ur, dotted, "\Gamma_2,\, A\leq T_1:K_1"'] &
\end{tikzcd}$$

so $All(A{\leq}T_1{:}K_1)T_2 \longrightarrow_{\Gamma_2} All(A{\leq}T_1{:}K_1)T_2''$. The result $\Gamma_2 \vdash All(A{\leq}T_1{:}K_1)T_2' \equiv All(A{\leq}T_1{:}K_1)T_2'' \in \star$ follows with E-All.

Subcase: $All(A{\leq}T_1{:}K_1)T_2 \longrightarrow_{\Gamma_1} All(A{\leq}T_1'{:}K_1)T_2$
where $T_1 \longrightarrow_{\Gamma_1} T_1'$. With the generation Lemmas 5.4 and 5.8 $\Gamma_1 \vdash T_1 \in K_1$ by subderivation, so by induction the exists a type T_1'' such that the following diagram commutes:

$$\begin{tikzcd}
T_1' \arrow[r, dotted, no head, "\Gamma_1", "K_1"'] & T_1'' \\
T_1. \arrow[u, "\Gamma_1"] \arrow[ur, dotted, "\Gamma_2"'] &
\end{tikzcd}$$

This implies $All(A{\leq}T_1{:}K_1)T_2 \longrightarrow_{\Gamma_2} All(A{\leq}T_1''{:}K_1)T_2$, and we conclude with E-All, E-Refl, and Lemma 5.32:

$$\frac{\Gamma_1 \vdash T_1' \equiv T_1'' \in K_1 \qquad \dfrac{\Gamma_1, A{\leq}T_1'{:}K_1' \vdash T_2 \in \star}{\Gamma_1, A{\leq}T_1'{:}K_1' \vdash T_2 \equiv T_2 \in \star}}{\Gamma_1 \vdash All(A{\leq}T_1'{:}K_1)T_2 \equiv All(A{\leq}T_1''{:}K_1)T_2 \in \star}$$

□

Proof of Lemma 5.67 on page 86: By induction on the length of derivation of $\Gamma \vdash S \equiv T \in K$. The case for E-Refl is immediate, the one for R-Top vacuously true. We show the ones for arrow-types, for All-types, and for application.

Case E-App+:
$$\frac{\Gamma \vdash_{\mathcal{A}} S_1 + \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1 \to K_2 \qquad \Gamma \vdash S_2 \equiv T_2 \in K_1}{\Gamma \vdash S_1\ S_2 \equiv T_1\ T_2 \in K_2}$$

By Definition A.10 of $\longrightarrow_{\Gamma}$ we know $S_1\ S_2$ must be of the form $A\ U_1 \ldots U_n$ for some

$n \geq 1$ and $S' = \Gamma(A)\ U_1 \dots U_n$; hence $T = A\ V_1 \dots V_n$ where $\Gamma \vdash A\ U_1 \dots U_{n-1} \equiv A\ V_1 \dots V_{n-1} \in K_1 \to K_2$ and $\Gamma \vdash U_n \equiv V_n \in K_1$. Thus we can choose $T' = \Gamma(A)\ V_1 \dots V_n$ (using preservation of kinding under reduction). The other application cases are analogous.

Case E-ARROW:
$$\frac{\Gamma \vdash T_1 \equiv S_1 \in \star \qquad \Gamma \vdash S_2 \equiv T_2 \in \star}{\Gamma \vdash S_1 \to S_2 \equiv T_1 \to T_2 \in \star}$$

The $\longrightarrow_\Gamma$-step can take two directions. For example $(S_1 \to S_2) \longrightarrow_\Gamma S'_1$ with $S_1 \longrightarrow_\Gamma S'_1$, and the case follows by generation for kinds, induction, and E-ARROW.

Case E-ALL:
$$\frac{\Gamma,\ A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1}{\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \equiv \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

We have further to distinguish, where the reduction step takes place:

Subcase: $\mathit{All}(A{\leq}S_1{:}K_1)S_2 \longrightarrow_\Gamma \mathit{All}(A{\leq}S'_1{:}K_1)S_2$
with $S_1 \longrightarrow_\Gamma S'_1$. By induction we get a type T'_1 with $T_1 \longrightarrow_\Gamma T''_1$ and $\Gamma \vdash T_1 \equiv T'_1 \in K_1$. Thus $\mathit{All}(A{\leq}T_1{:}K_1)T_2 \longrightarrow_\Gamma \mathit{All}(A{\leq}T'_1{:}K_1)T_2$ and the case follows with E-ALL.

Subcase: $\mathit{All}(A{\leq}S_1{:}K_1)S_2 \longrightarrow_\Gamma \mathit{All}(A{\leq}S_1{:}K_1)S'_2$
with $S_2 \longrightarrow_{(\Gamma,\ A\leq S_1:K_1)} S'_2$. By induction we $T_2 \longrightarrow_{(\Gamma,\ A\leq S_1:K_1)} U'_2$ and $\Gamma,\ A{\leq}S_1{:}K_1 \vdash S'_2 \equiv U'_2 \in K$ for some type U'. From $\Gamma,\ A{\leq}S_1{:}K_1 \equiv \Gamma,\ A{\leq}T_1{:}K_1$ we can use Lemma 5.66 to exchange the context in the step to get $T_2 \longrightarrow_{(\Gamma,\ A\leq S_1:K_1)} T''_2$ for some type T'', with $\Gamma,\ A{\leq}S_1{:}K \vdash U''_2 \equiv T''_2 \in \star$, so by transitivity of equivalence and E-ALL the result follows. □

Proof of Lemma 5.68 on page 87: By induction on $\Gamma \vdash T \in K$, using the definition of $\longrightarrow_\Gamma$, preservation of kinding under promotion, and Lemma 5.32.
□

Proof of Lemma 5.70 on page 88: By the previous Lemmas 5.69 and 5.68. □

C.6 Characterization of strong, cut-free derivations

Proof of Corollary 5.73 on page 89: With Lemma 5.40 for normalization with respect to $\beta\top$-reduction, Lemma 5.66 for the change of context, and for the Γ-steps Lemma 5.67 . □

Proof of Lemma 5.74 on page 90: By induction on the length of the reduction sequence $\nearrow^*_\Gamma = (\longrightarrow^!_{\beta\top} \uparrow_\Gamma)^n$.

For $n = 0$, we have $S = A\ S_1 \dots S_n$, since S must be in normal form, and the case is immediate. Since for a type U with $U \longrightarrow_{\beta\top} U'$ or $U \uparrow_\Gamma U'$ we know $U\ T \longrightarrow_{\beta\top} U'\ T$ respectively $U\ T \uparrow_\Gamma U'\ T$, the induction step can be proven by an inner induction on the number of $\beta\top$-steps resp. the $\uparrow_\Gamma$-step. □

Proof of Lemma 5.75 on page 90: By definition, $\nearrow^*_\Gamma$ consists of a sequence of $\longrightarrow_{\beta\top}$– and $\uparrow_\Gamma$–steps and the result follows by a sequence of instances of the promotion rule, well-kindedness of subderivations, and the expansion lemma. □

Proof of Lemma 5.76 on page 90: By induction on the length of derivation, using the generation Lemma 5.8, and the subtyping rules for application. □

Proof of Lemma 5.77 on page 90: By induction on the derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$.

Using strong derivations and in absence of transitivity, only the rules of reflexivity, of promotion, or one of the application rules are usable.

Case R-Refl:
$$\frac{S \longrightarrow^!_{\beta\top} A\; U_1 \ldots U_n \qquad T \longrightarrow^!_{\beta\top} A\; U_1 \ldots U_n \qquad \Gamma \vdash A\; U_1 \ldots U_n \in K}{\Gamma \vdash_{\mathcal{CS}} S \leq T \in K}$$

By reflexivity of equivalence or reflexivity of subtyping, depending on the minimal polarity of the type $A\; U_1 \ldots U_i$ in Γ.

Case R-Promote:
$$\frac{S \longrightarrow^!_{\beta\top} U \uparrow_\Gamma S' \qquad \Gamma \vdash U \in K \qquad \Gamma \vdash_{\mathcal{CS}} S' \leq T \in K}{\Gamma \vdash_{\mathcal{CS}} S \leq T \in K}$$

By preservation of kinding under reduction (Lemma 5.25) and under promotion (Lemma 5.45), S' is well-kinded in context Γ, and the case follows by induction.

Case R-App$\circ_r$:
$$\frac{\begin{array}{c} S \longrightarrow^!_{\beta\top} S'_1\; S'_2 \qquad T \longrightarrow^!_{\beta\top} A\; T_1 \ldots T_n \\ \Gamma \vdash_{\mathcal{CS}} S'_1 \leq A\; T_1 \ldots T_{n-1} \in K_1 \to K_2 \qquad \Gamma \vdash_{\mathcal{A}} A\; T_1 \ldots T_{n-1} \circ \end{array}}{\Gamma \vdash_{\mathcal{CS}} S \leq T \in K_2}$$

By well-kinded subderivations, S'_1 is well-kinded in Γ. So by induction $S'_1 \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\; S_1 \ldots S_{n-1}$ and additionally for all $0 \leq i \leq n-1$, one of the four listed cases holds. Now since $\Gamma \vdash_{\mathcal{A}} T_1 \ldots T_{n-1} \circ$, we get with Lemma 5.76 also $\Gamma \vdash A\; S_1 \ldots S_{n-1} \circ$, which solves the case for $i = n$. Finally with Lemma 5.74 $S \longrightarrow^!_{\beta\top} S'_1\; S'_2 \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A\; S_1 \ldots S_{n-1}\; S'_2$.

The other application cases are similar. The additional requirements in the cases 2 and 3 concerning the subderivation hold in these cases by induction and (for S_n in relation to T_n) by the unique choice of the application rule. □

Proof of Lemma 5.78 on page 91: Similar to the proof for applications in Lemma 5.77. The first part by induction over the derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T \in \star$. Only the rules R-Refl, R-Promote, and R-Arrow are applicable. For reflexivity the case is immediate. The case for promotion follows by induction. Rule R-Arrow finally expresses directly the content of part 1 of the lemma. The second and the third part are similar, with rules R-Refl, R-Promote and R-All, resp. R-Abs. □

Proof of Lemma 5.79 on page 91: We know $S \nearrow_\Gamma^* \longrightarrow_{\beta\top}^! A\ S_1 \dots S_n$ by Lemma 5.77. By well-kindedness of subderivation, preservation of kinding under promotion and under reduction (Lemma 5.45 and Lemma 5.25) $\Gamma \vdash A\ S_1 \dots S_n \in K'$. With the help of Lemma 5.76 also $\Gamma \vdash A\ U_1 \dots U_n \in K'$. □

Proof of Lemma 5.80 on page 91: By induction on n, using Lemma 5.72, generation for kinds, and preservation of kinding under promotion. For $n = 0$ the case is immediate by the generation lemma, preservation of kinding under promotion and reflexivity. For $n > 0$, we get by induction $\Gamma \vdash_{\mathcal{CS}} \Gamma(A)\ S_1 \dots S_{n-1} \leq \Gamma(A)\ T_1 \dots T_{n-1} \in K_1 \to K_2$. In case 2, for example, we get by preservation of kinding under promotion $\Gamma \vdash_{\mathcal{A}} \Gamma(A)\ S_1 \dots S_{n-1}\ +$ or $\Gamma \vdash_{\mathcal{A}} \Gamma(A)\ S_1 \dots S_{n-1}\ \circ$. Since $\Gamma \vdash_{\mathcal{CS}} S_n \leq T_n \in K_n$, we get using R-App$+_l$ or R-App$+_r$, or one of the rules for constant application that $\Gamma \vdash_{\mathcal{C}} \Gamma(A)\ S_1 \dots S_{n-1}\ S_n \leq \Gamma(A)\ T_1 \dots T_{n-1}\ T_n \in K$ and the result $\Gamma \vdash_{\mathcal{CS}} \Gamma(A)\ S_1 \dots S_{n-1}\ S_n \leq \Gamma(A)\ T_1 \dots T_{n-1}\ T_n \in K$ follows by strengthening for cut-free derivations (Lemma 5.72). □

Proof of Lemma 5.81 on page 92: By induction on the length of the sequence $T \nearrow_\Gamma^* \longrightarrow_{\beta\top}^! A\ T_1 \dots T_m$, i.e., on the length of

$$T\ (\longrightarrow_{\beta\top}^! \uparrow_\Gamma)^n \longrightarrow_{\beta\top}^! A\ T_1 \ \dots T_m,$$

using the characterization of subtypes of an application (Lemma 5.77).

If $n = 0$, we are given $T \longrightarrow_{\beta\top}^! A\ T_1 \dots T_m$. So Lemma 5.77 applies, yielding directly the result. For $n \geq 1$ are given

$$T \longrightarrow_{\beta\top}^! \uparrow_\Gamma U\ (\longrightarrow_{\beta\top}^! \uparrow_\Gamma)^{n-1} \longrightarrow_{\beta\top}^! A\ T_1 \ \dots T_m.$$

By definition of promotion $T \longrightarrow_{\beta\top}^! A'\ U_1 \dots U_k \uparrow_\Gamma \Gamma(A')\ U_1 \dots U_k = U$. By preservation of kinding under reduction and promotion, U is well-kinded. From $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$ and $T \longrightarrow_{\beta\top}^! A'\ U_1 \dots U_k$, we get by the characterization of subtypes of an application (Lemma 5.77):

$$S \nearrow_\Gamma^* \longrightarrow_{\beta\top}^! A'\ V_1 \dots V_k \quad \text{and} \quad \Gamma \vdash_{\mathcal{CS}} A'\ V_1 \dots V_k \leq A'\ U_1 \dots U_k \in K'$$

Moreover, all V_i and U_i are related according to the kind of A as given by the lemma, which means that Lemma 5.80 applies, yielding $\Gamma \vdash_{\mathcal{CS}} \Gamma(A')\ V_1 \dots V_k \leq \Gamma(A')\ U_1 \dots U_k \in K$, i.e. $\Gamma \vdash_{\mathcal{CS}} \Gamma(A')\ V_1 \dots V_k \leq U \in K$. Thus by induction $\Gamma(A')\ V_1 \dots V_k \nearrow_\Gamma^* \longrightarrow_{\beta\top}^! A\ S_1 \dots S_m$ with the postulated connection for the S_i and T_i. Note that by construction $S \nearrow_\Gamma^* \longrightarrow_{\beta\top}^! A\ S_1 \dots S_m$, contains at least one promotion step. □

Proof of Lemma 5.82 on page 92: Similar to the proof for applications in Lemma 5.81, using Lemma 5.78. □

C.7 Cut elimination

Proof of cut elimination (Proposition 5.83 on page 94): By induction on the combined length of derivation. We have to generalize the induction hypothesis slightly to cope with a pair of two All-rules, which extend the context by two different, albeit equivalent, upper bounds:

> Suppose $\vdash \Gamma_1 \equiv \Gamma_2$ and the types S, T, U and V well-kinded in context Γ_1 (and thus by Lemma 5.32 also in Γ_2). Furthermore assume U and V in normal form. If $\Gamma_1 \vdash_{\mathcal{CS}} S \leq U \in K$ and $\Gamma_2 \vdash_{\mathcal{CS}} V \leq T \in K$ with $\Gamma_1 \vdash U \equiv V \in K$, then $\Gamma_1 \vdash_{\mathcal{CS}} S \leq T \in K$ (and $\Gamma_2 \vdash_{\mathcal{CS}} S \leq T \in K$).

The proposition follows from this generalization because of reflexivity of equivalence and since $\Gamma \vdash_{\mathcal{CS}} T_1 \leq T_2 \in K$ iff. $\Gamma \vdash_{\mathcal{CS}} T_1^! \leq T_2^! \in K$. We proceed by case analysis on the syntactic structure of T's normal form.

Case: $T \longrightarrow^!_{\beta\top} \mathit{Top}(K')$
By well-kindedness of subderivations (Lemma 5.49) we get $\Gamma \vdash \mathit{Top}(K') \in K$, $\Gamma \vdash U \in K$, and $\Gamma \vdash S^! \in K$; thus the case follows by S-Top.

Case: $T \longrightarrow^!_{\beta\top} T_1 \to T_2$
By the characterization of subtypes in Lemma 5.78 we get $V \nearrow^*_{\Gamma_2} \longrightarrow^!_{\beta\top} V_1 \to V_2$ with $\Gamma_2 \vdash_{\mathcal{CS}} T_1 \leq V_1 \in \star$ and $\Gamma_2 \vdash_{\mathcal{CS}} T_2 \leq V_2 \in \star$, both by subderivation of $\Gamma_2 \vdash_{\mathcal{CS}} V \leq T \in \star$. From the equivalences $\Gamma_1 \vdash U \equiv V \in \star$ and $\vdash \Gamma_1 \equiv \Gamma_2$ we get by Corollary 5.73 a type $U_1 \to U_2$ with $U \nearrow^*_{\Gamma_1} \longrightarrow^!_{\beta\top} U_1 \to U_2$ and $\Gamma_1 \vdash U_1 \to U_2 \equiv V_1 \to V_2 \in \star$. Now by Lemma 5.82 $S \nearrow^*_{\Gamma_1} \longrightarrow^!_{\beta\top} S_1 \to S_2$ with $\Gamma_1 \vdash_{\mathcal{CS}} U_1 \leq S_1 \in \star$ and $\Gamma_1 \vdash_{\mathcal{CS}} S_2 \leq U_2 \in \star$, justified by subderivation of $\Gamma_1 \vdash_{\mathcal{CS}} S \leq U \in \star$. Hence by induction $\Gamma_1 \vdash_{\mathcal{CS}} T_1 \leq S_1 \in \star$ and $\Gamma_1 \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$, yielding $\Gamma_1 \vdash_{\mathcal{CS}} S_1 \to S_2 \leq T_1 \to T_2 \in \star$ by R-Arrow. By Lemma 5.75 we get $\Gamma_1 \vdash_{\mathcal{CS}} S \leq T \in \star$.

Case: $T \longrightarrow^!_{\beta\top} \mathit{All}(A{\leq}T_1{:}K_1)T_2$
Analogously to the previous case $V \nearrow^*_{\Gamma_2} \longrightarrow^!_{\beta\top} \mathit{All}(A{\leq}V_1{:}K_1)V_2$ such that

$$\frac{\begin{array}{c} T \longrightarrow^!_{\beta\top} \mathit{All}(A{\leq}T_1{:}K)T_2 \quad \Gamma_2 \vdash V_1 \equiv T_1 \in K_1 \\ \Gamma_2,\, A{\leq}V_1{:}K_1 \vdash_{\mathcal{CS}} V_2 \leq T_2 \in \star \end{array}}{\Gamma_2 \vdash_{\mathcal{CS}} \mathit{All}(A{\leq}V_1{:}K_1)V_2 \leq T \in \star}$$

and where $\Gamma_2, A{\leq}V_1{:}K_1 \vdash_{\mathcal{CS}} V_2 \leq T_2 \in \star$ is justified by a subderivation of $\Gamma_2 \vdash_{\mathcal{CS}} V \leq T \in \star$. By Corollary 5.73, the two equivalences $\Gamma_2 \vdash U \equiv V \in \star$ and $\vdash \Gamma_1 \equiv \Gamma_2$ entail $U \nearrow^*_{\Gamma_1} \longrightarrow^!_{\beta\top} \mathit{All}(A{\leq}U_1{:}K_1)U_2$ with $\Gamma_1 \vdash \mathit{All}(A{\leq}U_1{:}K_1)U_2 \equiv \mathit{All}(A{\leq}V_1{:}K_1)V_2 \in \star$, which also means $\Gamma_1 \vdash U_1 \equiv V_1 \in K_1$ and $\Gamma_1, A{\leq}U_1{:}K_1 \vdash U_2 \leq V_2 \in \star$. Now Lemma 5.82 yields $S \nearrow^*_{\Gamma_1} \longrightarrow^!_{\beta\top} \mathit{All}(A{\leq}S_1{:}K_1)S_2$ with

$$\Gamma_1 \vdash S_1 \equiv U_1 \in K_1 \quad \text{and} \quad \Gamma_1,\, A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2 \leq U_2 \in \star,$$

where the subtype statement is justified by a subderivation of $\Gamma_1 \vdash_{\mathcal{CS}} S \leq U \in \star$.

By transitivity of equivalence $\Gamma_1 \vdash_{\mathcal{CS}} S_1 \equiv T_1 \in \star$ and $\Gamma_1,\ A{\leq}S_1{:}K_1 \vdash U_2 \equiv V_2 \in \star$, and by the definition of equivalence on contexts $\vdash \Gamma_1,\ A{\leq}S_1{:}K_1 \equiv \Gamma_2,\ A{\leq}V_1{:}K_1$. Thus by induction $\Gamma_1,\ A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$, yielding $\Gamma_1 \vdash_{\mathcal{CS}} \mathit{All}(A{\leq}S_1{:}K_1)S_1 \leq \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star$ by R-All. Finally by Lemma 5.75, we get $\Gamma_1 \vdash_{\mathcal{CS}} S \leq T \in \star$.

Case: $T \longrightarrow^!_{\beta\top} \mathit{Fun}(A{:}K)T'$
Similar.

Case: $T \longrightarrow^!_{\beta\top} A'\ T_1 \ldots T_m$
for some $m \geq 1$. By Lemma 5.77 we obtain $V \nearrow^*_{\Gamma_2} \longrightarrow^!_{\beta\top} A'\ V_1 \ldots V_m$. Additionally, for all $i \in \{0, \ldots m\}$, one of the following cases holds (where V_0 and T_0 stand for A'):

1. $\Gamma_2 \vdash A'\ V_1 \ldots V_{i-1} \circ$ and $\Gamma_2 \vdash_{\mathcal{A}} A'\ T_1 \ldots T_{i-1} \circ$.
2. $\Gamma_2 \vdash_{\mathcal{A}} A'\ V_1 \ldots V_{i-1} +$ and $\Gamma_2 \vdash A'\ T_1 \ldots T_{i-1} +$, with $\Gamma_2 \vdash_{\mathcal{CS}} V_i \leq T_i \in K_i$.
3. $\Gamma_2 \vdash_{\mathcal{A}} A'\ V_1 \ldots V_{i-1} -$ and $\Gamma_2 \vdash A'\ T_1 \ldots T_{i-1} -$, with $\Gamma_2 \vdash_{\mathcal{CS}} T_i \leq V_i \in K_i$.
4. $\Gamma_2 \vdash V_i \equiv T_i \in K_i$.

As in the previous subcases we use Corollary 5.73 to obtain $U \nearrow^*_{\Gamma_1} A'\ U_1 \ldots U_m$ with $\Gamma_1 \vdash A'\ U_1 \ldots U_m \equiv A'\ V_1 \ldots V_m \in K$. By Lemma 5.81, we get $S \nearrow^*_{\Gamma_1} \longrightarrow^!_{\beta\top} A'\ S_1 \ldots S_m$ with the S_i in appropriate connection with the corresponding U_i, i.e. for all $i \in \{0, \ldots m\}$, one of the following cases holds:

a) $\Gamma_1 \vdash_{\mathcal{A}} A'\ S_1 \ldots S_{i-1} \circ$ and $\Gamma_1 \vdash A'\ S_1 \ldots S_{i-1} \circ$.

b) $\Gamma_1 \vdash_{\mathcal{A}} A'\ S_1 \ldots S_{i-1} +$ and $\Gamma_1 \vdash A'\ U_1 \ldots U_{i-1} +$, with $\Gamma_1 \vdash_{\mathcal{CS}} S_i \leq U_i \in K_i$.

c) $\Gamma_1 \vdash_{\mathcal{A}} A'\ S_1 \ldots S_{i-1} -$ and $\Gamma_1 \vdash A'\ U_1 \ldots U_{i-1} -$, with $\Gamma_1 \vdash_{\mathcal{CS}} U_i \leq S_i \in K_i$.

d) $\Gamma_1 \vdash_{\mathcal{A}} S_i \equiv U_i \in K_i$.

Since the choice of case is determined by the minimal polarity of the corresponding type operator, we can use Lemma 5.76 and Lemma 5.55 to infer that the cases 1 – 4 correspond to the cases a) – d) in such a way that if case 1) holds for U_i and T_i, then correspondingly case a) holds for S_i and U_i. Hence we can use induction in case 2 in combination with b), and case 3 in combination with c) to obtain $\Gamma \vdash_{\mathcal{CS}} S_i \leq T_i \in K_i$ respectively $\Gamma \vdash_{\mathcal{CS}} T_i \leq S_i \in K_i$. Case 4/d) can be treated by transitivity of equivalence. The constant case is immediate. We thus get $\Gamma_1 \vdash_{\mathcal{CS}} A'\ S_1 \ldots S_m \leq A'\ U_1 \ldots U_m \in K$. By Lemma 5.75 finally $\Gamma_1 \vdash_{\mathcal{CS}} S \leq T \in K$. $\square$

C.8 Elimination of promotion

Proof of Proposition 5.84 on page 95: For the proof, we assume, that the types S and T are in normal form. Since $\Gamma \vdash_{\mathcal{CS}} U_1 \leq U_2 \in K$ iff. $\Gamma \vdash_{\mathcal{CS}} U_1^! \leq U_2^! \in K$, this is not a restriction. For types in normal form, the fact that $\Gamma \vdash_{\mathcal{CS}} S \gtrless T \in K$ has derivations without using promotion means $\Gamma \vdash S \equiv T \in K$. Thus, the implication we are to show reads

> If $\Gamma \vdash_{\mathcal{CS}} S \leq T \in K$ and $\Gamma \vdash_{\mathcal{CS}} T \leq S \in K$ with S and T in normal form, then $\Gamma \vdash S \equiv T \in K$.

The proof will proceed by induction on the sum of ranks (Definition 5.71) of both statements.

If one of the two subtyping derivation ends with an instance of R-Refl, we have by uniqueness of normal forms $S = T$ and the case follows by reflexivity of equivalence. So let us assume, none of the derivations ends with an instance of reflexivity.

Case: $\Gamma \vdash_{\mathcal{CS}} S_1 \to S_2 \leq T_1 \to T_2 \in \star$ and $\Gamma \vdash_{\mathcal{CS}} T_1 \to T_2 \leq S_1 \to S_2 \in \star$
We get $\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1 \in \star$ and $\Gamma \vdash_{\mathcal{CS}} S_1 \leq T_1 \in \star$, two statements whose sum of ranks is strictly smaller than the sum of ranks of the two original statements. Hence by induction $\Gamma \vdash T_1 \equiv S_1 \in \star$. Symmetrically $\Gamma \vdash S_2 \equiv T_2 \in \star$, so by E-Arrow $\Gamma \vdash S_1 \to S_2 \equiv T_1 \to T_2 \in \star$.

Case: $\Gamma \vdash_{\mathcal{CS}} All(A{\leq}S_1{:}K_1)S_2 \leq All(A{\leq}T_1{:}K_1)T_2 \in \star$ and
$\Gamma \vdash_{\mathcal{CS}} All(A{\leq}T_1{:}K_1)T_2 \leq All(A{\leq}S_1{:}K_1)S_2 \in \star$
We get by subderivation $\Gamma \vdash_{\mathcal{CS}} T_1 \leq S_1 \in K_1$ and $\Gamma \vdash_{\mathcal{CS}} S_1 \leq T_1 \in K_1$ as well as $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$ and $\Gamma, A{\leq}T_1{:}K_1 \vdash_{\mathcal{CS}} T_2 \leq S_2 \in \star$. By induction further $\Gamma \vdash S_1 \equiv T_1 \in K_1$.

The rank of $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{CS}} S_2 \leq T_2 \in \star$ is strictly less than the rank of $\Gamma \vdash_{\mathcal{CS}} All(A{\leq}S_1{:}K_1)S_2 \leq All(A{\leq}T_1{:}K_1)T_2 \in \star$. Likewise the rank of $\Gamma, A{\leq}T_1{:}K_1 \vdash_{\mathcal{CS}} T_2 \leq S_2 \in \star$ compared to the one of $\Gamma \vdash_{\mathcal{CS}} All(A{\leq}T_1{:}K_1)T_2 \leq All(A{\leq}S_1'{:}K_1)S_2' \in \star$ (note that $\vdash \Gamma, A{\leq}S_1{:}K_1 \equiv \Gamma, A{\leq}T_1{:}K_1$). Hence again by induction $\Gamma_1, A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star$ which, together with $\Gamma \vdash S_1 \equiv T_1 \in K_1$, implies the result using E-All.

Case: $\Gamma \vdash_{\mathcal{CS}} A_1\, S_1 \ldots S_n \leq A_2\, T_1 \ldots T_m \in K$ and $\Gamma \vdash_{\mathcal{CS}} A_2\, T_1 \ldots T_m \leq A_1\, S_1 \ldots S_n \in K$
(The form of the applications with a leading type variable on each side is justified by the assumption that S and T are in normal form.) By Lemma 5.77 we get two types $A_2\, S_1' \ldots S_m'$ and $A_1\, T_1' \ldots T_n'$ with

$$A_1\, S_1 \ldots S_n \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A_2\, S_1' \ldots S_m' \quad \text{and} \quad A_2\, T_1 \ldots T_m \nearrow^*_\Gamma \longrightarrow^!_{\beta\top} A_1\, T_1' \ldots T_n'.$$

Moreover, the lemma implies that for all $i \in \{1 \dots m\}$ at least one of the following cases holds:

$$\begin{aligned}
&\Gamma \vdash_{\mathcal{A}} A_2\ S'_1 \dots S'_{i-1} \circ && && \text{(C.1)}\\
&\Gamma \vdash_{\mathcal{A}} A_2\ S'_1 \dots S'_{i-1} + &&\text{and}\quad \Gamma \vdash_{\mathcal{CS}} S'_i \leq T_i \in K_i && \text{(C.2)}\\
&\Gamma \vdash_{\mathcal{A}} A_2\ S'_1 \dots S'_{i-1} - &&\text{and}\quad \Gamma \vdash_{\mathcal{CS}} T_i \leq S'_i \in K'_i && \text{(C.3)}\\
& && \phantom{\text{and}}\quad \Gamma \vdash S'_i \equiv T_i \in K'_i && \text{(C.4)}
\end{aligned}$$

A dual condition holds for all T'_j in relation with S_j where $j \in \{1, \dots n\}$.

By Lemma 5.75, the sequence $A_1\ S_1 \dots S_n \nearrow^{*}_{\Gamma} \longrightarrow^{!}_{\beta\top} A_2\ S'_1 \dots S'_m$ entails $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \dots S_n \leq A_2\ S'_1 \dots S'_m \in K'$ for some kind K'. Since $A_1\ S_1 \dots S_n$ is well-kinded and in normal form, we get from $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \dots S_n \leq A_2\ T_1 \dots T_m \in K$ by well-kindedness of subderivations $\Gamma \vdash A_1\ S_1 \dots S_n \in K$ and further by preservation of kinding under promotion and reduction $\Gamma \vdash A_2\ S'_1 \dots S'_m \in K$; thus by Lemma 5.53 $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \dots S_n \leq A_2\ S'_1 \dots S'_m \in K$. Since $A_2\ S'_1 \dots S'_n$ is well-kinded, we get by cut-elimination (Proposition 5.83) $\Gamma \vdash_{\mathcal{CS}} A_2\ T_1 \dots T_m \leq A_2\ S'_1 \dots S'_m \in K$. Dually, we have $\Gamma \vdash_{\mathcal{CS}} A_2\ T_1 \dots T_m \leq A_1\ T'_1 \dots T'_n \in K$ and again by cut-elimination $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \dots S_n \leq A_1\ T'_1 \dots T'_n \in K$.

We now distinguish, whether the sequences $A_1\ S_1 \dots S_n \nearrow^{*}_{\Gamma} \longrightarrow^{!}_{\beta\top} A_2\ S'_1 \dots S'_m$ and $A_2\ T_1 \dots T_m \nearrow^{*}_{\Gamma} \longrightarrow^{!}_{\beta\top} A_1\ T'_1 \dots T'_n$ are empty or not.

Subcase: $A_1\ S_1 \dots S_n \nearrow^{+}_{\Gamma} \longrightarrow^{!}_{\beta\top} A_2\ S'_1 \dots S'_m$ and $A_2\ T_1 \dots T_m \nearrow^{+}_{\Gamma} \longrightarrow^{!}_{\beta\top} A_1\ T'_1 \dots T'_n$

where both sequences are non-empty. Now the rank of $\Gamma \vdash_{\mathcal{CS}} A_2\ S'_1 \dots S'_m \leq A_2\ T_1 \dots T_m \in K$ is strictly smaller that the one of $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \dots S_n \leq A_2\ T_1 \dots T_m \in K$.

Likewise the rank of $\Gamma \vdash_{\mathcal{CS}} A_1\ T'_1 \dots T'_n \leq A_1\ S_1 \dots S_n \in K$ has properly decreased compared to $\Gamma \vdash_{\mathcal{CS}} A_2\ T_1 \dots T_m \leq A_1\ S_1 \dots S_n \in K$.

So we have arrived at the situation (the following abbreviates four subtyping statements):

$$\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \dots S_n \leq A_2\ S'_1 \dots S'_m \leq A_2\ T_1 \dots T_m \leq A_1\ T'_1 \dots T'_n \leq A_1\ S_1 \dots S_n \in K$$

Projecting out the pair $\Gamma \vdash_{\mathcal{CS}} A_2\ S'_1 \dots S'_m \leq A_2\ T_1 \dots T_m \in K$ and $\Gamma \vdash_{\mathcal{CS}} A_2\ T_1 \dots T_n \leq A_2\ S'_1 \dots S'_m \in K$ (using cut-elimination and the fact that all types involved are well-kinded), we observe that the sum of ranks of this pair of statements has properly decreased, hence by induction $\Gamma \vdash A_2\ S'_1 \dots S'_m \equiv A_2\ T_1 \dots T_m \in K$. By a dual argument $\Gamma \vdash A_1\ S_1 \dots S_n \equiv A_1\ T'_1 \dots T'_m \in K$. Finally also the sum of ranks of the pair $\Gamma \vdash_{\mathcal{CS}} A_1\ T'_1 \dots T'_n \gtrless A_2\ S'_1 \dots S'_n \in K$ has properly decreased in comparison to the original pair of statements, hence once more by induction $\Gamma \vdash A_1\ T'_1 \dots T'_n \equiv A_2\ S'_1 \dots S'_n \in K$. By transitivity of equivalence thus

$\Gamma \vdash A_1\ S_1 \ldots S_n \equiv A_2\ T_1 \ldots T_m \in K$ (which additionally implies $A_1 = A_2$ and $n = m$). This means

$$\begin{aligned} A_1\ S_1 \ldots S_n \nearrow^+_\Gamma \ &\longrightarrow^!_{\beta\top} A_1\ S'_1 \ldots S'_m \equiv^K_\Gamma \\ &\equiv^K_\Gamma A_1\ T_1 \ldots T_m \nearrow^+_\Gamma \ \longrightarrow^!_{\beta\top} A_1\ T'_1 \ldots T'_n \equiv^K_\Gamma A_1\ S_1 \ldots S_n. \end{aligned}$$

This contradicts strong termination of $\longrightarrow_{\beta\top\Gamma\equiv} \equiv$ (Lemma 5.69), and the subcase cannot occur.

Subcase: $A_1\ S_1 \ldots S_n = A_2\ S'_1 \ldots S'_m$ and $A_2\ T_1 \ldots T_m \nearrow^+_\Gamma \ \longrightarrow^!_{\beta\top} A_1\ T'_1 \ldots T'_n$ (i.e. $A_1 = A_2$ and $n = m$) where $A_1\ T_1 \ldots T_n \nearrow^+_\Gamma A_1\ T'_1 \ldots T'_n$ by a non-empty sequence. We are thus given the following three subtyping statements:

$$\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \ldots S_n \leq A_1\ T_1 \ldots T_n \leq A_1\ T'_1 \ldots T'_n \leq A_1\ S_1 \ldots S_n \in K.$$

Arguing in the same way as in the previous subcase we get by induction $\Gamma \vdash A_1\ S_1 \ldots S_n \equiv A_1\ T'_1 \ldots T'_n \in K$. We are not done, though, since we cannot compare the pair of statements $\Gamma \vdash_{\mathcal{CS}} A_1\ T_1 \ldots T_n \gtreqless A_1\ S'_1 \ldots S'_n \in K$ to the original pair $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \ldots S_n \gtreqless A_1\ T_1 \ldots T_n \in K$ with respect to their sum of ranks.

But now $A_1\ T_1 \ldots T_1 \nearrow^+_\Gamma \ \longrightarrow^!_{\beta\top} A_1\ T'_1 \ldots T'_n$ together with the four the conditions (C.1) – (C.4) from above imply $A_1\ S_1 \ldots S_n \nearrow^+_\Gamma \ \longrightarrow^!_{\beta\top} A_1\ S''_1 \ldots S''_n$ (Lemma 5.81) for some kind $A_1\ S''_1 \ldots S''_n$ by a non-empty sequence. Moreover, this lemma entails $\Gamma \vdash_{\mathcal{CS}} A_1\ S''_1 \ldots S''_n \leq A_1\ T'_1 \ldots T'_n \in K$. Putting it all together we get the following chain of subtyping statements:

$$\begin{aligned} \Gamma \vdash_{\mathcal{CS}} A_1\ T_1 \ldots T_n \leq A_1\ S_1 \ldots S_n &\leq A_1\ S''_1 \ldots S''_n \\ &\leq A_1\ T'_1 \ldots T'_n \leq A_1\ S_1 \ldots S_n \leq A_1\ T_1 \ldots T_n \in K. \end{aligned}$$

Cut-elimination yields the two statements $\Gamma \vdash_{\mathcal{CS}} A_1\ T_1 \ldots T_n \gtreqless A_1\ S''_1 \ldots S''_n \in K$ and $\Gamma \vdash_{\mathcal{CS}} A_1\ T'_1 \ldots T'_n \gtreqless A_1\ S''_1 \ldots S''_n \in K$. By induction and transitivity of equivalence $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \ldots S_n \equiv A_1\ T_1 \ldots T_n \in K$.

This means we are given

$$\begin{aligned} A_1\ T_1 \ldots T_n \equiv^K_\Gamma A_1\ S_1 \ldots S_n \nearrow^+_\Gamma A_1\ S'_1 \ldots S'_n &\equiv^K_\Gamma \\ \equiv^K_\Gamma A_1\ T'_1 \ldots T'_n \equiv^K_\Gamma A_1\ S_1 \ldots S_n &\equiv^K_\Gamma A_1\ T_1 \ldots T_n, \end{aligned}$$

which again contradicts Lemma 5.69, and neither this subcase can occur.

Subcase: $A_1\ S_1 \ldots S_n = A_2\ S'_1 \ldots S'_m$ and $A_2\ T_1 \ldots T_m = A_1\ T'_1 \ldots T'_n$
Thus we assume both sequences empty. Again, $A_1 = A_2$ and $n = m$. So we are given $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \ldots S_n \leq A_1\ T_1 \ldots T_n \in K$ and $\Gamma \vdash_{\mathcal{CS}} A_1\ S_1 \ldots S_n \leq A_1\ T_1 \ldots T_n \in K$. Now that for all $i \in \{1, \ldots, n\}$ we know $S_i = S'_i$ and $T_i = T'_i$, we get by the four conditions (C.1) – (C.4) from above that for all i at least one of the following cases holds:

1. $\Gamma \vdash_{\mathcal{A}} A_1\ S_1 \ldots S_{i-1} \circ$.
2. $\Gamma \vdash_{\mathcal{A}} A_1\ S_1 \ldots S_{i-1} +$ and $\Gamma \vdash_{\mathcal{CS}} S_i \gtrless T_i \in K_i$.
3. $\Gamma \vdash_{\mathcal{A}} A_1\ S_1 \ldots S_{i-1} -$ and $\Gamma \vdash_{\mathcal{CS}} T_i \gtrless S_i \in K_i$.
4. $\Gamma \vdash T_i \equiv S_i \in K_i$.

Thus, in the cases 2 and 3 we get by induction $\Gamma \vdash S_i \equiv T_i \in K_i$, and the result follows by the appropriate number of instances of application rules for equivalence.

□

C.9 Decidability of polarized subtyping

C.9.1 Completeness

We first prove that derivability in the subtyping system implies well-kindedness of the types involved. The two parts of the lemma are analogous to Lemma 5.49, to Lemma 5.52, and to Lemma 5.53 for the stratified versions of the subtyping system. The first part is stronger than Lemma 5.49, since for the stratified subtyping system we could prove only that well-kindedness of types in a subtyping statement is propagated to all subtyping statements justified by a subderivation. The extra assumption of well-kindedness was needed there, since expansion may destroy well-kindedness. The lemma will be needed for the proof of completeness and soundness.

Lemma C.2

1. If $\Gamma \vdash_{\mathcal{O}} S \leq T \in K$, then $\Gamma \vdash_{\mathcal{O}} S \in K$ and $\Gamma \vdash_{\mathcal{O}} \in K$.
2. If $\Gamma \vdash_{\mathcal{O}} S \leq T \in K$, and $K \leq K'$, then $\Gamma \vdash_{\mathcal{O}} S \leq T \in K'$.
3. If $\Gamma \vdash_{\mathcal{O}} S \leq T \in K$, with $\Gamma \vdash_{\mathcal{O}} S \in K'$ and $\Gamma \vdash_{\mathcal{O}} T \in K'$, then $\Gamma \vdash_{\mathcal{O}} S \leq T \in K'$.

Proof: By straightforward induction. □

Proof of Lemma 5.86 on page 96: By straightforward induction. □

Lemma C.3 Let Γ abbreviate Γ_1, $A_2{:}K'$, $A_1{\leq}A_2{:}K'$, Γ_2 or Γ_1, $A_2{:}K'$, $A_1{:}K'$, Γ_2. If $\Gamma \vdash [A_1/A]T \equiv [A_2/A]T \in K$, then $[A/A_1][A/A_2]\Gamma \vdash T \circ_A$.

Proof: By induction over the length of derivation, using the properties of polarities.

□

Proof of Lemma 5.87 on page 96: By induction on the length of derivation. We show the first part of the lemma; the rest is similar. Let Γ abbreviate the context Γ_1, $A_2{:}K'$, $A_1{\leq}A_2{:}K'$, Γ_2.

Case R-Refl: $$\frac{[A_1/A]T \longrightarrow^{!}_{\beta\top} U \qquad [A_2/A]T \longrightarrow^{!}_{\beta\top} U \qquad \Gamma \vdash U \in K}{\Gamma \vdash_{\mathcal{CS}} [A_1/A]T \leq [A_2/A]T \in K}$$

Since $A_1 \neq A_2$ and furthermore neither A_1 nor A_2 occurs freely in T, we know $A_1 \notin fv(U)$ and $A_2 \notin fv(U)$. This also means $A \notin fv(T^!)$, implying with the help of preservation of kinding under reduction that $\Gamma \vdash T^! \circ_A$, and hence also $\Gamma \vdash T^! +_A$.

Case R-Promote: $$\frac{\begin{array}{c}[A_1/A]T \longrightarrow^{!}_{\beta\top} U \uparrow_\Gamma T' \\ \Gamma \vdash_{\mathcal{CS}} U \in K \qquad \Gamma \vdash_{\mathcal{CS}} T' \leq [A_2/A]T \in K\end{array}}{\Gamma \vdash_{\mathcal{CS}} [A_1/A]T \leq [A_2/A]T \in K}$$

By definition of promotion and the properties of reduction

$$[A_1/A]T \longrightarrow^{!}_{\beta\top} A'\ [A_1/A]T_1 \dots [A_1/A]T_n \uparrow_\Gamma \Gamma(A')\ [A_1/A]T_1 \dots [A_1/A]T_n = T'.$$

We have to distinguish whether or not A' and A_1 coincide.

Subcase: $A' = A_1$
In this case the subgoal of the promotion rule reads

$$\Gamma \vdash_{\mathcal{CS}} A_2\ [A_1/A]T_1 \dots [A_1/A]T_n \leq [A_2/A]T \in K.$$

Note, that $A_2\ [A_1/A]T_1 \dots [A_1/A]T_n$ is in normal form. Here, the derivation cannot end in an instance of R-Promote, since A_2's upper bound is a maximal type; hence the last rules applied must be either instances of one of the application rules or of reflexivity. Thus we get for all $i \in \{1, \dots, n\}$ one of the following cases (we have $\Gamma \vdash_{\mathcal{A}} A_2\ [A_1/A]T_1 \dots [A_1/A]T_{i-1}\ ?$ iff. $\Gamma \vdash_{\mathcal{A}} A_2\ [A_2/A]T_1 \dots [A_2/A]T_{i-1}\ ?$ by Lemma 5.76), so we need not consider the symmetric cases where the polarity is determined by the right-hand side):

1. $\Gamma \vdash_{\mathcal{A}} A_2\ [A_1/A]T_1 \dots [A_1/A]T_{i-1}\ \circ$.
2. $\Gamma \vdash_{\mathcal{A}} A_2\ [A_1/A]T_1 \dots [A_1/A]T_{i-1}\ +$ and $\Gamma \vdash_{\mathcal{CS}} [A_1/A]T_i \leq [A_2/A]T_i \in K_i$.
3. $\Gamma \vdash_{\mathcal{A}} A_2\ [A_1/A]T_1 \dots [A_1/A]T_{i-1}\ -$ and $\Gamma \vdash_{\mathcal{CS}} [A_2/A]T_i \leq [A_1/A]T_i \in K_i$.
4. $\Gamma \vdash_{\mathcal{A}} A_2\ [A_1/A]T_1 \dots [A_1/A]T_{i-1}\ \pm$ and $\Gamma \vdash [A_2/A]T_i \equiv [A_1/A]T_i \in K_i$.

The mentioned subtyping statements in case 3 and 4 are justified by a subderivation. Thus the result follows by induction, by an inner induction on $n \geq 1$, Lemma 5.86, and the rules for polarity. In the case for $\Gamma \vdash [A_1/A]T_i \equiv [A_2/A]T_i \in K_i$, we additionally use the corresponding for lemma for equivalence (Lemma C.3.), yielding $\Gamma \vdash T_i \circ_A$.

Subcase: $A' \neq A_1$
By the previous Lemma 5.86 we have $\Gamma' \vdash_{\mathcal{CS}} [A/A_1][A/A_2](\Gamma(A'))\ T_1 \ldots T_n \leq T \in K$, which means $\Gamma' \vdash_{\mathcal{CS}} \Gamma'(A')\ T_1 \ldots T_n \leq T \in K$. By design of the $\vdash_{\mathcal{CS}}$-system with its normalizing reductions, this statement is derivable iff. $\Gamma' \vdash_{\mathcal{CS}} \Gamma'(A')\ T_1 \ldots T_n \leq T^! \in K$ is derivable, i.e. $\Gamma' \vdash_{\mathcal{CS}} \Gamma'(A')\ T_1 \ldots T_n \leq A'\ T_1 \ldots T_n \in K$ (the types T_i are in normal form already). Since we know by reflexivity of subtyping $\Gamma' \vdash A'\ T_1 \ldots T_n \gtrless A'\ T_1 \ldots T_n \in K$, a derivation of $\Gamma' \vdash_{\mathcal{CS}} \Gamma'(A')\ T_1 \ldots T_n \leq A'\ T_1 \ldots T_n \in K$ thus contradicts Lemma 5.84 and the subcase cannot occur.

Case R-All:
$$\frac{\begin{array}{c}[A_1/A]T \xrightarrow{!}_{\beta\top} [A_1/A](All(A'{\leq}T_1{:}K_1)T_2) \\ [A_2/A]T \xrightarrow{!}_{\beta\top} [A_2/A](All(A'{\leq}T_1{:}K_1)T_2) \\ \Gamma \vdash_{\mathcal{CS}} [A_1/A]T_1 \equiv [A_2/A]T_1 \in K_1 \\ \Gamma, A'{\leq}[A_1/A]T_1{:}K_1 \vdash_{\mathcal{CS}} [A_1/A]T_2 \leq [A_2/A]T_2 \in \star\end{array}}{\Gamma \vdash_{\mathcal{CS}} [A_1/A]T \leq [A_2/A]T \in \star}$$

By Lemma C.3, $\Gamma' \vdash T_1 \circ_A$. By Definition 5.38 $\vdash \Gamma', A'{\leq}T_1{:}K_1 \circ_A$, hence the induction hypothesis applies to $\Gamma, A'{\leq}[A_1/A]T_1{:}K_1 \vdash_{\mathcal{CS}} [A_1/A]T_2 \leq [A_2/A]T_2 \in \star$, as well, yielding $\Gamma', A'{\leq}T_1{:}K_1 \vdash T_2 +_A$, thus

$$\frac{\Gamma', A'{\leq}T_1{:}K_1 \vdash T_2 +_A \qquad \Gamma' \vdash T_1 \circ_A}{\Gamma' \vdash All(A'{\leq}T_1{:}K_1)T_2 +_A}$$

Case R-App$+_l$:
$$\frac{\begin{array}{c}[A_1/A]T \xrightarrow{!}_{\beta\top} [A_1/A](T_1\ T_2) \qquad [A_2/A]T \xrightarrow{!}_{\beta\top} [A_2/A](T_1\ T_2) \\ \Gamma \vdash_{\mathcal{A}} [A_1/A]T_1 + \quad \Gamma \vdash_{\mathcal{CS}} [A_1/A]T_1 \leq [A_2/A]T_1 \in K_1 \to K_2 \\ \Gamma \vdash_{\mathcal{CS}} [A_1/A]T_2 \leq [A_2/A]T_2 \in K_1\end{array}}{\Gamma \vdash_{\mathcal{CS}} [A_1/A]T \leq [A_2/A]T \in K_2}$$

By induction $\Gamma' \vdash T_1 +_A$ and $\Gamma' \vdash T_2 +_A$ (both T_1 and T_2 are in normal form), and the result follows from the application rule for polarities. The remaining cases are similar. □

C.9.2 Soundness

Proof of Lemma 5.91 on page 99: By induction on the length of derivation in each part.

We start with part 1 for contexts. The case for the empty context is immediate, the one for term variables is solved by straightforward induction.

Case C-TVar:
$$\frac{A' \notin dom(\Gamma'') \qquad \Gamma'' \vdash_{\mathcal{O}} T \in K''}{\vdash_{\mathcal{O}} \Gamma'', A'{\leq}T{:}K''\ ok}$$

i.e. $\Gamma = \Gamma'', A'{\leq}T{:}K''$. We have to distinguish whether or not the type variable A'

coincides with one of the A_i's. If it does, we are given more specifically $\Gamma = \Gamma^l_i, A_i{:}K_i$ and:

$$\frac{A \notin dom(\Gamma^l_i) \qquad \Gamma^l_i \vdash_{\mathcal{O}} Top(K_i) \in K_i}{\vdash_{\mathcal{O}} \Gamma^l_i, A_i{:}K_i\ ok}$$

If for instance $\Gamma' = \Gamma^{l'}_i, A''_i{:}K_i, A'_i{\leq}A''_i{:}K_i$, we can solve the case by C-TVar and K-TVar (remember that $\Gamma'^{l}_i, A''_i{:}K_i$ abbreviates $\Gamma'^{l}_i, A''_i{\leq}Top(K_i){:}K_i$):

$$\frac{A'_i \notin dom(\Gamma^{l'}_i, A''_i{:}K_i) \qquad \dfrac{\dfrac{A''_i \notin dom(\Gamma^{l'}_i) \qquad \Gamma^{l'}_i \vdash_{\mathcal{O}} Top(K_i) \in K_i}{\vdash_{\mathcal{O}} \Gamma^{l'}_i, A''_i{:}K_i\ ok}}{\Gamma^{l'}_i, A''_i{:}K_i \vdash_{\mathcal{O}} A''_i \in K_i}}{\vdash_{\mathcal{O}} \Gamma^{l'}_i, A''_i{:}K_i, A'_i{\leq}A''_i{:}K_i\ ok}$$

The statement on the top follows by induction on part 2. If Γ' is of one of the other possible forms, the case is similar.

If $A' \neq A_i$ for some i, the induction hypothesis of part 2 applies yielding $\Gamma'' \vdash_{\mathcal{O}} T' \in K$.

Next the cases for kinding in part 2. The subsumption rule is solved by induction; likewise the two rules for the maximal types. We show the case for All-types; the ones for arrow-types and for K-Arrow-E are simpler.

Case K-All:
$$\frac{\Gamma, A'{\leq}T_1{:}K_1 \vdash_{\mathcal{O}} T_2 \in \star}{\Gamma \vdash_{\mathcal{O}} All(A'{\leq}T_1{:}K_1)T_2 \in \star}$$

By induction $\Gamma', A'{\leq}T'_1{:}K_1 \vdash_{\mathcal{O}} T'_2 \in \star$, and the case follows by K-All.

Case K-TVar:
$$\frac{kind_{\Gamma}A' = K \qquad \vdash_{\mathcal{O}} \Gamma\ ok}{\Gamma \vdash_{\mathcal{O}} A' \in K}$$

If $A' = A_i$ for some i, then $K = K_i$, and $T' = A'_i$ or $T' = A''_i$. In both cases the result follows by induction on part 1 of the lemma and K-TVar. If $A' \neq A_i$ for all A_i, the result follows similarly.

Case K-Arrow-I+:
$$\frac{\begin{array}{c}\Gamma, A_0{:}K_0 \vdash_{\mathcal{O}} S \in K'_0 \\ \Gamma, A''_0{:}K_0, A'_0{\leq}A''_0{:}K_0 \vdash_{\mathcal{O}} [A'_0/A_0]S \leq [A''_0/A_0]S \in K'_0\end{array}}{\Gamma \vdash_{\mathcal{O}} Fun(A_0{:}K_0)S \in K_0 \rightarrow^{+} K'_0}$$

By assumption $[\vec{A}/\vec{A''}][\vec{A}/\vec{A''}]T' = T = Fun(A_0{:}K_0)S$; hence $T' = Fun(A_0{:}K_0)S'$ with $[\vec{A}/\vec{A'}][\vec{A}/\vec{A''}]S' = S$. Induction on part 2 gives $\Gamma', A''_0{:}K_0, A'_0{\leq}A''_0{:}K_0 \vdash_{\mathcal{O}} S' \in K'_0$.

From the second subgoal $\Gamma, A''_0{:}K_0, A'_0{\leq}A''_0{:}K_0 \vdash_{\mathcal{CS}} [A'_0/A_0]S^{!} \leq [A''_0/A_0]S^{!} \in K'_0$ by completeness (Proposition 5.88), thus by Lemma 5.87 $\Gamma, A_0{:}K_0 \vdash S^{!} +_{A_0}$. This means we can apply the induction hypothesis of part 4 to the kinding subgoal (For the $A_1, \dots, A_n$ case 4 d) holds, for the additional variable A_0 part 4 b) of the lemma applies.), yielding $\Gamma', A_0'':K_0, A_0' \leq A_0'':K_0 \vdash_{\mathcal{O}} [A_0'/A_0]S' \leq [A_0''/A_0]S' \in K_0'$, and the case follows by K-Arrow-I+.

Case K-Arrow-I∘:

$$\frac{\begin{array}{c}\Gamma, A_0:K_0 \vdash_{\mathcal{O}} S \in K_0' \\ \Gamma, A_0'':K_0, A_0':K_0 \vdash_{\mathcal{O}} [A_0'/A_0]S \leq [A_0''/A_0]S \in K_0'\end{array}}{\Gamma \vdash_{\mathcal{O}} \mathit{Fun}(A_0:K_0)S \in K_0 \to^{\circ} K_0'}$$

By the generation lemma and induction $\Gamma', A':K_0 \vdash_{\mathcal{O}} S' \in K_0'$. Analogously to the previous case, the result follows by induction on part 4, completeness (Proposition 5.88), Lemma 5.87, and K-Arrow-I∘. The remaining antimonotone and nonmonotone case for K-Arrow-I± and K-Arrow-I− are similar.

The cases for part 3, dealing with the occurrence of type variables, by straightforward induction.

Finally the cases of part 4, the more interesting part. Proceed by induction on the derivation of $\Gamma \vdash_{\mathcal{O}} T \in K$. The cases for K-Top⋆ and K-Top are solved by induction. Likewise the ones for K-Arrow and K-All, where for universally quantified types we need additionally the generation lemma for polarities to check that the extended context conforms to the conditions in part a) – c) of the lemma.

Case K-Subsumption:

$$\frac{K' \leq K \qquad \Gamma \vdash_{\mathcal{O}} T \in K'}{\Gamma \vdash_{\mathcal{O}} T \in K}$$

By induction on part 4, $\Gamma' \vdash_{\mathcal{O}} T\sigma_1 \leq T\sigma_2 \in K'$, and the case follows by Lemma C.2.

Case K-TVar:

$$\frac{K = \mathit{kind}_{\Gamma} A' \qquad \vdash_{\mathcal{O}} \Gamma\ \mathit{ok}}{\Gamma \vdash_{\mathcal{O}} A' \in K}$$

We have to distinguish whether or not A' coincides with one of the A_i's. If it does not, the case follows by S-Conv, rule K-TVar, and induction on part 1:

$$\cfrac{\cfrac{K = \mathit{kind}_{\Gamma'} A' \qquad \vdash_{\mathcal{O}} \Gamma'\ \mathit{ok}}{\Gamma' \vdash_{\mathcal{O}} A' \in K}}{\Gamma' \vdash_{\mathcal{O}} A' \leq A' \in K}$$

If $A_i = A'$ for some i and case d) holds for A_i, we are given $K = \mathit{kind}_{\Gamma'} A_i = K_i$ and $\sigma_1(A_i) = \sigma_2(A_i)$. Thus the case follows by S-Conv and K-TVar.

If $A_i = A'$ for some i and one of the cases a) – c) holds we know that Γ' must be of the form $\Gamma'^{l}_{i}, A_i'':K_i, A_i' \leq A_i'':K_i, \Gamma'^{r}_{i}$, since $\Gamma \vdash A_i +_{A_i}$ but neither the constant case a) nor the negative one c) applies. Furthermore $\sigma_1(A_i) = A_i'$ and $\sigma_2(A_i) = A_i''$. The goal $\Gamma' \vdash_{\mathcal{O}} A_i' \leq A_i'' \in K_i$ follows by transitivity, S-Conv, K-TVar, and induction on part 1 of the lemma.

Case K-ARROW±:
$$\frac{\Gamma,\, A_0{:}K_1 \vdash_{\mathcal{O}} T \in K_2}{\Gamma \vdash_{\mathcal{O}} \mathit{Fun}(A_0{:}K_1)T \in K_1 \rightarrow^{\pm} K_2}$$

By the generaton lemma (respectively Observation 5.90) for polarities and induction on part 4, we get $\Gamma',\, A_0{:}K_1 \vdash_{\mathcal{O}} T\sigma_1 \leq T\sigma_2 \in K_2$, and thus conclude by S-ARROW±:

$$\frac{\Gamma',\, A_0{:}K_1 \vdash_{\mathcal{O}} T\sigma_1 \leq T\sigma_2 \in K_2}{\Gamma' \vdash_{\mathcal{O}} \mathit{Fun}(A_0{:}K_1)T\sigma_1 \leq \mathit{Fun}(A_0{:}K_1)T\sigma_2 \in K_1 \rightarrow^{\pm} K_2}$$

Case K-ARROW-I+:
$$\frac{\begin{array}{c}\Gamma,\, A_0{:}K_1 \vdash T \in K_2 \\ \Gamma,\, A_0''{:}K_1,\, A_0'{\leq}A_0''{:}K_1 \vdash [A_0'/A_0]T \leq [A_0''/A_0]T \in K_1\end{array}}{\Gamma \vdash \mathit{Fun}(A_0{:}K_1)T \in K_1 \rightarrow^{+} K_2}$$

By induction and using the generation lemma $\Gamma',\, A_0''{:}K_1,\, A_0'{\leq}A_0''{:}K_1 \vdash [A_0'/A_0]T\sigma_1 \leq [A_0''/A_0]T\sigma_2 \in K_1$ and the case follows by rule K-ARROW-I+. The cases for K-ARROW-I− and K-ARROW∘ are solved similarly.

Case K-ARROW-E:
$$\frac{\Gamma \vdash_{\mathcal{O}} T_1 \in K_1 \rightarrow^{?} K_2 \qquad \Gamma \vdash_{\mathcal{O}} T_2 \in K_1}{\Gamma \vdash_{\mathcal{O}} T_1\; T_2 \in K_2}$$

By the generation lemma for kinding, $\Gamma \vdash T_1\; T_2\;\; ?'_{A_i}$ implies $\Gamma \vdash T_1\; ?'_{A_i}$ for all variables A_i. Thus, the induction hypothesis applies yielding $\Gamma' \vdash_{\mathcal{O}} [\vec{A}'/\vec{A}]T_1 \leq [\vec{A}''/\vec{A}]T_1 \in K_1 \rightarrow^{?} K_2$.

For the argument T_2, we distinguish according to the minimal polarity of T_1 as type operator. If, for example, $\Gamma \vdash_{\mathcal{A}} T_1\; \pm$, we get for all A_i by Lemma 5.21 either $\Gamma \vdash_{\mathcal{O}} T_2\; \circ_{A_i}$, if one of the cases a) – c) holds, or we know $\sigma_1(A_i) = \sigma_2(A_i)$. Further by twice induction $\Gamma' \vdash_{\mathcal{O}} T_2\sigma_1 \gtrless T_2\sigma_2 \in K_1$, so by transitivity, S-APP, and S-APP±:

$$\frac{\dfrac{\Gamma' \vdash_{\mathcal{O}} T_2\sigma_1 \gtrless T_2\sigma_2 \in K_1}{\Gamma' \vdash_{\mathcal{O}} (T_1\; T_2)\sigma_1 \leq T_1\sigma_1\; T_2\sigma_2 \in K_2} \qquad \dfrac{\Gamma' \vdash_{\mathcal{O}} T_1\sigma_1 \leq T_1\sigma_2 \in K_1 \rightarrow^{?} K_2}{\Gamma' \vdash_{\mathcal{O}} T_1\sigma_1\; T_2\sigma_2 \leq (T_1\; T_2)\sigma_2 \in K_2}}{\Gamma' \vdash_{\mathcal{O}} (T_1\; T_2)\sigma_1 \leq (T_1\; T_2)\sigma_2 \in K_2}$$

The other cases are similar. □

Lemma C.4 If $\Gamma \vdash T \in K$ with T in normal form and $T \Uparrow_{\Gamma} T'$, then $\Gamma \vdash_{\mathcal{O}} T \leq T' \in K$.

Lemma C.5 Assume a well-formed context $\Gamma = \Gamma_1,\, A{\leq}U{:}K_1,\, \Gamma_2$.

1. Let $\Gamma' = \Gamma_1,\, A{\leq}V{:}K_1,\, \Gamma_2$ and assume $\Gamma_1 \vdash_{\mathcal{O}} V \leq U \in K_1$.

   (a) If $\vdash_{\mathcal{O}} \Gamma$ *ok* , then $\vdash_{\mathcal{O}} \Gamma'$ *ok*.

(b) If $\Gamma \vdash_{\mathcal{O}} T \in K$, then $\Gamma' \vdash_{\mathcal{O}} T \in K$.

(c) If $\Gamma \vdash_{\mathcal{O}} S \leq T \in K$, then $\Gamma' \vdash_{\mathcal{O}} S \leq T \in K$.

2. Assume $\Gamma_1 \vdash U \in K_1'$ with $K_1' \leq K_1$ and let $\Gamma' = \Gamma_1, A{\leq}V{:}K_1, \Gamma_2$.

   (a) If $\vdash_{\mathcal{O}} \Gamma$ *ok* , then $\vdash_{\mathcal{O}} \Gamma'$ *ok*.

   (b) If $\Gamma \vdash_{\mathcal{O}} T \in K$, then $\Gamma' \vdash_{\mathcal{O}} T \in K$.

   (c) If $\Gamma \vdash_{\mathcal{O}} S \leq T \in K$, then $\Gamma' \vdash_{\mathcal{O}} S \leq T \in K$.

Proof: By straightfoward induction, using well-kindedness of subderivations, weakening for subtyping and for kinding. □

Proof of soundness (Lemma 5.92 on page 99): By induction on the length of derivation with the help of part 4 of the previous Lemma 5.91 for the case of arrow introduction.

The first part dealing with the contexts by straightforward induction, as the rules in both formulations coincide.

The cases for kinding statements of the form $\Gamma \vdash T \in K$ are in most cases also straightforward. The interesting case is the one for arrow introduction.

Case K-ARROW-I?:

$$\frac{\Gamma, A{:}K_1 \vdash T \ ?_A \qquad \Gamma, A{:}K_1 \vdash T \in K_2}{\Gamma \vdash \mathit{Fun}(A{:}K_1)T \in K_1 \rightarrow^{?} K_2}$$

By induction we get $\Gamma, A{:}K_1 \vdash_{\mathcal{O}} T \in K_2$. We distinguish according to the polarity of A inside T. If $? = \pm$ the case follows by K-ARROW$\pm$ of the original system. If $? = +$, we get by induction on the second part of the lemma $\Gamma, A{:}K_1 \vdash_{\mathcal{O}} T +_A$. Hence by Lemma 5.91 $\Gamma, A_2{:}K_1, A_1{\leq}A_2{:}K_1 \vdash_{\mathcal{O}} [A_1/A]T \leq [A_2/A]T \in K$ and we can conclude by K-ARROW-I+:

$$\frac{\Gamma, A_2{:}K_1, A_1{\leq}A_2{:}K_1 \vdash_{\mathcal{O}} [A_1/A]T \leq [A_2/A]T \in K_2 \qquad \Gamma, A{:}K_1 \vdash_{\mathcal{O}} T \in K_2}{\Gamma \vdash_{\mathcal{O}} \mathit{Fun}(A{:}K_1)T \in K_1 \rightarrow^{+} K_2}$$

The fourth part for equivalence by straightforward induction. E-REFL follows by induction on part 3 of the lemma and S-CONV, E-TOP likewise by induction and S-TOP.

Case E-All:
$$\frac{\Gamma, A{\leq}S_1{:}K_1 \vdash S_2 \equiv T_2 \in \star \qquad \Gamma \vdash S_1 \equiv T_1 \in K_1}{\Gamma \vdash \mathit{All}(A{\leq}S_1{:}K_1)S_2 \equiv \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star}$$

By induction we get $\Gamma, A{\leq}S_1{:}K_1 \vdash_{\mathcal{O}} S_2 \gtreqless T_2 \in \star$ and $\Gamma \vdash_{\mathcal{O}} S_1 \gtreqless T_1 \in K_1$. This means by S-All $\Gamma \vdash_{\mathcal{O}} \mathit{All}(A{\leq}S_1{:}K_1)S_2 \leq \mathit{All}(A{\leq}T_1{:}K_1)T_2 \in \star$. Since $\Gamma \vdash_{\mathcal{O}} T_1 \leq S_1 \in K_1$, we get from Lemma C.5 that also $\Gamma, A{\leq}T_1{:}K_1 \vdash_{\mathcal{O}} S_2 \gtreqless T_2 \in \star$, from which again with S-All second subtyping statement $\Gamma \vdash_{\mathcal{O}} \mathit{All}(A{\leq}T_1{:}K_1)T_2 \leq \mathit{All}(A{\leq}S_1{:}K_1)S_2 \in \star$ follows. E-Arrow is easier. E-Abs by induction and Lemma C.5.

Part 5 dealing with the subtyping statements is again straightforward. The case for R-Refl follows by well-kindedness of subderivations (Lemma 5.49), induction on the kinding part of the lemma, and S-Conv. R-Promote is solved by preservation of kinding under promotion, Lemma C.4, induction on part 3 and part 5 of the lemma, and S-Trans. R-Top follows directly from S-Top. R-Arrow and R-All by induction, where for the upper bounds of the universally quantified types we use induction over part 4 of the lemma. The application cases also by induction, using an additional instance of transitivity to compensate for the fact, that the application rules in the original system are more strict in that the insist on the to type operators to be identical. In case of R-App$+_l$, for instance, we get from $\Gamma \vdash_{\mathcal{CS}} S_1\ S_2 \leq T_1\ T_2 \in K_2$ where $S \longrightarrow^{!}_{\beta\top} S_1\ S_2$ and $T \longrightarrow^{!}_{\beta\top} T_1\ T_2$ by induction on the subtyping subgoals $\Gamma \vdash_{\mathcal{O}} S_1 \leq T_1 \in K_1 \rightarrow^{+} K_2$ and $\Gamma \vdash_{\mathcal{O}} S_2 \leq T_2 \in K_1$. Thus we can derive with transitivity, S-App, and S-App+:

$$\frac{\dfrac{\Gamma \vdash_{\mathcal{O}} S_1 \in K_1 \rightarrow^{+} K_2 \quad \Gamma \vdash_{\mathcal{O}} S_2 \leq T_2 \in K_1}{\Gamma \vdash_{\mathcal{O}} S_1\ S_2 \leq S_1\ T_2 \in K_2} \qquad \dfrac{\Gamma \vdash_{\mathcal{O}} S_1 \leq T_1 \in K_1 \rightarrow K_2}{\Gamma \vdash_{\mathcal{O}} S_1\ T_2 \leq T_1\ T_2 \in K_2}}{\Gamma \vdash_{\mathcal{O}} S_1\ S_2 \leq T_1\ T_2 \in K_2}$$

The remaining application cases are similar, and R-Abs again by straightforward induction, using well-kindedness of subderivations. □

C.10 Typing

Proof of Lemma 6.1 on page 105: By induction on the length of the typing derivation. The case for T-Subsumption is immediate. In case of of a term variable (rule T-TVar) the result follows the generation lemma for contexts (Lemma 5.4). Arrow-elimination by straightforward induction, using the kinding rules for arrow-types. The case for arrow-introduction is similar, using additionally the generation lemma for contexts. The rules for All-types again by induction, using the generation Lemma 5.8 for kinds and the kinding rules for universally quantified types. □

Proof of Corollary 6.3: On well-kinded inputs, the original subtyping relation is equivalent to strong, cut-free derivations (soundness and completeness of 5.92 and 5.88). Proceed by induction on a derivation of $\Gamma \vdash_{\mathcal{CS}} S \leq T_1 \to T_2$. Thus the result follows by Lemma 5.78. $\square$

Fact C.6

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

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

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

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

3a. If S is a minimal type for s in Γ and $S \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} T_1 \to T_2$ and T is a minimal type for t in Γ with $\Gamma \vdash T \leq T_1 \in \star$, then T_2 is a minimal type for $s\ t$ in Γ.

3b. If s or t has no type in Γ, or if S is a minimal type for s in Γ and T is a minimal type for t in Γ but $S \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} S' \neq T_1 \to T_2$, or if $S \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} S' = T_1 \to T_2$ but $\Gamma \not\vdash T \leq T_1 \in \star$, then $s\ t$ has no type in Γ.

4a. If T_2 is a minimal type of e in Γ, $A{\leq}T_1{:}K_1$, then $\mathit{All}(A{\leq}T_1{:}K_1)T_2$ is a minimal type of $\mathit{fun}(A{\leq}T_1{:}K_1)e$ in Γ.

4b. If e has no type in Γ, $A{\leq}T_1{:}K_1$, then $\mathit{fun}(A{\leq}T_1{:}K_1)e$ has no type in Γ.

5a. If S is a minimal type for s in Γ and $S \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} \mathit{All}(A{\leq}T_1{:}K_1)T_2$ and $\Gamma \vdash U \leq T_1 \in K_1$, then $[T/A]T_2$ is a minimal type for $s\ U$ in Γ.

5b. If s has no type in Γ or if S is a minimal type for s in Γ but $S \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} S' \neq \mathit{All}(A{\leq}T_1{:}K_1)T_2$ or if $S \nearrow^{!}_{\Gamma} \longrightarrow^{!}_{\beta\top} \mathit{All}(A{\leq}T_1{:}K_1)T_2$ but $\Gamma \not\vdash U \leq T_1 \in K_1$, then $s\ U$ has no type in Γ.

Proof of Theorem 6.6: By induction, using the previous facts. $\square$