

Typability is Undecidable for F+Eta*

J. B. Wells

jbw@cs.bu.edu

+1 617 739 7456, FAX +1 617 353 6457

Boston Univ. Dept. of Computer Science

Boston, MA 02215, U.S.A.

March 9, 1996

Abstract

System F is the well-known polymorphically-typed λ -calculus with universal quantifiers (“ \forall ”). F+ η is System F extended with the eta rule, which says that if term M can be given type τ and M η -reduces to N , then N can also be given the type τ . Adding the eta rule to System F is equivalent to adding the subsumption rule using the subtyping (“containment”) relation that Mitchell defined and axiomatized [Mit88]. The subsumption rule says that if M can be given type τ and τ is a subtype of type σ , then M can be given type σ . Mitchell’s subtyping relation involves no extensions to the syntax of types, i.e., no bounded polymorphism and no supertype of all types, and is thus unrelated to the system F_< (“F-sub”).

Typability for F+ η is the problem of determining for any term M whether there is any type τ that can be given to it using the type inference rules of F+ η . Typability has been proven undecidable for System F [Wel94] (without the eta rule), but the decidability of typability has been an open problem for F+ η . Mitchell’s subtyping relation has recently been proven undecidable [TU95, Wel95b], implying the undecidability of “type checking” for F+ η . This paper reduces the problem of subtyping to the problem of typability for F+ η , thus proving the undecidability of typability. The proof methods are similar in outline to those used to prove the undecidability of typability for System F, but the fine details differ greatly.

1 Introduction

1.1 Background and Motivation Girard [Gir72] and Reynolds [Rey74] independently formulated the type system of the second-order, parametrically-polymorphic λ -calculus about twenty years ago. Girard developed his system (named by chance “System F”) to prove properties of second-order propositional logic (hence F’s other name, “the second-order λ -calculus”) while Reynolds wanted to express polymorphic typing in programming explicitly. Both Girard and Reynolds formulated F in the “Church style”, but we deal with the “Curry style” formulation first given by Leivant [Lei83]. In the Church style, types are embedded in terms and the term-formation rules are also the typing rules, while in the Curry style, types are given to pure terms of the λ -calculus. The inference rules of a type system derive statements called *sequents* of the form “ $A \vdash M : \tau$ ”, where A is a set of type assumptions for free variables, the subject M is a term, and the predicate τ is a type that can be given to M . For a type system in the Curry style, it becomes meaningful to ask for an arbitrary λ -term M :

*This work is partly supported by NSF grants CCR-9113196 and CCR-9417382.

1. Is there any typing for M , i.e., do there exist assumptions A and type τ such that $A \vdash M : \tau$ is derivable?
2. Can M be given some particular type τ using some particular type assumptions A , i.e., for arbitrarily chosen A and τ , is $A \vdash M : \tau$ derivable?

The first problem is named *typability* and the second *type checking*.

Since typability and type checking for System F are both undecidable [Wel94], it is natural to examine extensions and restrictions of F in the hope of finding a type system for which typability or type checking is decidable. One way of extending a type system is to require typings to be closed under some relation on untyped λ -terms. If R is a binary relation on untyped λ -terms, then we can extend a type system with the typing rule (R):

$$\frac{A \vdash M : \tau, \quad R(M, N)}{A \vdash N : \tau} \quad (R)$$

Of course, there are many interesting relations which are worth considering. Another way of extending a type system is to add a subtyping relation. If “ \leq ” is a binary relation on types, then we can extend a type system with the *subsumption* rule:

$$\frac{A \vdash M : \sigma, \quad \sigma \leq \tau}{A \vdash M : \tau} \quad (\text{subsum})$$

In this paper we will consider a type system which can be defined as an extension of System F using either of these methods.

The system $F+\eta$ extends System F with a rule that allows η -reducing the subject of a sequent. System F does not already have the subject η -reduction property, so an additional type inference rule is necessary. The rule (\rightarrow_η) defined according to the pattern for (R) above is actually more general than necessary [Mit90]; the following more restricted rule is sufficient:

$$\frac{A \vdash \lambda x.Mx : \sigma \rightarrow \tau}{A \vdash M : \sigma \rightarrow \tau} \quad x \notin \text{FV}(M) \quad (\eta)$$

$F+\eta$ is then defined to be System F extended by rule (η). It will be described below how this extension is equivalent to adding a particular subtyping relation.

Every typing in F is also a typing in $F+\eta$, but there are terms typable in $F+\eta$ that are not typable in F and particular terms can also be given more types. For example, this sequent:

$$\{x : \forall \alpha. (\alpha \rightarrow \alpha)\} \vdash x : (\forall \alpha. \alpha) \rightarrow ((\forall \alpha. \alpha) \rightarrow (\forall \alpha. \alpha))$$

is derivable in $F+\eta$ but not in F. The corresponding η -expansion which F can handle is:

$$\{x : \forall \alpha. (\alpha \rightarrow \alpha)\} \vdash \lambda z.xz : (\forall \alpha. \alpha) \rightarrow ((\forall \alpha. \alpha) \rightarrow (\forall \alpha. \alpha))$$

Thus, the term $(\lambda x.x)$ can be given the type $(\forall \alpha. (\alpha \rightarrow \alpha)) \rightarrow ((\forall \alpha. \alpha) \rightarrow ((\forall \alpha. \alpha) \rightarrow (\forall \alpha. \alpha)))$ in $F+\eta$ but not in F.

Adding the (η) rule to System F turns out to have a model-theoretic justification. First, we will present the background and then we will explain the connection.

Mitchell devised a notion of a *type inference model* for System F to explain the assignment of types to terms [Mit90]. In these models, the meaning of a typing statement “ $M : \tau$ ” is that the meaning of the term M belongs to a set of λ -term meanings associated with the meaning of the

type τ , which is written as $\llbracket M \rrbracket \in D_{[\tau]}$.¹ In a type inference model, the meaning of a term M of type $\tau \rightarrow \rho$ must be a function that maps any term meaning of type τ to a term meaning of type ρ :

$$\llbracket M \rrbracket \in D_{[\tau \rightarrow \rho]} \implies (\llbracket M \rrbracket \cdot D_{[\tau]}) \subseteq D_{[\rho]} \quad (1)$$

However, if the meaning of a term M happens to be a function that maps any term meaning of type σ to a term meaning of type τ , it is not required that M be of type $\sigma \rightarrow \tau$, so the implication in equation (1) only goes in one direction. In the reverse direction is the weaker requirement

$$(\llbracket M \rrbracket \cdot D_{[\tau]}) \subseteq D_{[\rho]} \implies \varepsilon \cdot \llbracket M \rrbracket \in D_{[\tau \rightarrow \rho]} \quad (2)$$

where ε is a distinguished element in the model satisfying certain requirements. It is quite natural to consider requiring type inference models to satisfy the following strengthening of (1):

$$\llbracket M \rrbracket \in D_{[\tau \rightarrow \rho]} \iff (\llbracket M \rrbracket \cdot D_{[\tau]}) \subseteq D_{[\rho]} \quad (3)$$

The stronger semantic restriction leads to fewer models existing. Fewer models result in more sound typings that are satisfied by all models.

Now we present the connection between the semantic considerations and $F+\eta$. It turns out that the distinguished model element ε is just $\llbracket \lambda x.\lambda y.xy \rrbracket$. Thus, (2) can be rewritten as:

$$(\llbracket M \rrbracket \cdot D_{[\tau]}) \subseteq D_{[\rho]} \implies \llbracket (\lambda x.\lambda y.xy)M \rrbracket \in D_{[\tau \rightarrow \rho]}$$

In $F+\eta$, the term M has some type σ if and only if the term $((\lambda x.\lambda y.xy)M)$ has type σ . Thus, $F+\eta$ satisfies the stronger requirement in equation (3). Mitchell showed that $F+\eta$ is also complete with respect to (3) because it derives the additional typings that are implied by (3) [Mit90].²

Mitchell proved that adding the (η) rule to System F is equivalent to adding the subsumption rule with a particular subtyping relation. One way of viewing subtyping that explains its close connection to η -reduction is that subtyping can be performed by the insertion of coercion functions which are $\beta\eta$ -equivalent or η -equivalent to the identity. By analyzing the effects of the (η) rule, Mitchell devised a subtyping rule system (which he called *containment*) such that adding the subsumption rule and his subtyping rules to System F had exactly the same effect as adding the (η) rule. These rules are given in Figure 1. This paper will use the name $F+\eta$ for System F extended by Mitchell's subtyping relation, since a sequent $A \vdash M : \tau$ is derivable in the former system if and only if it is derivable in the latter.

Adding Mitchell's subtyping relation involves no extensions to the syntax of types, i.e., no bounded polymorphism and no supertype of all types. It has little in common with the subtyping relation of the system F_{\leq} , which was proven undecidable by Pierce [Pie92]. In F_{\leq} , quantifiers have bounds, e.g., $\forall \alpha \leq \sigma.\tau$, and the subtyping relation is not closed under a rule for quantifier instantiation. It is unknown if there is any relation between the decidability of the two kinds of subtyping.

There is an interesting semantic justification for why Mitchell used the name "containment" instead of "subtyping". In Mitchell's system, the syntactic construct $\sigma \leq \tau$ is derivable if and only if the corresponding semantic statement is true:

$$\sigma \leq \tau \iff D_{[\sigma]} \subseteq D_{[\tau]}$$

¹A type inference model can be seen as a PER model where each PER contains exactly one equivalence class, the largest possible. Thus, it can not explain equality between terms.

²The proof depends on adding the $(=_{\beta})$ rule because the models used identify β -equivalent terms.

$$\begin{array}{lll}
(\text{sub}) & \forall \vec{\alpha}. \sigma \leq \forall \vec{\beta}. (\sigma[\vec{\alpha} := \vec{\tau}]) & \vec{\beta} \notin \text{FTV}(\forall \vec{\alpha}. \sigma) \\
(\text{distr}) & \forall \vec{\alpha}. (\sigma \rightarrow \tau) \leq (\forall \vec{\alpha}. \sigma) \rightarrow (\forall \vec{\alpha}. \tau) \\
(\rightarrow) & \frac{\sigma_2 \leq \sigma_1, \quad \tau_1 \leq \tau_2}{\sigma_1 \rightarrow \tau_1 \leq \sigma_2 \rightarrow \tau_2} \\
(\text{trans}) & \frac{\rho \leq \sigma, \quad \sigma \leq \tau}{\rho \leq \tau} \\
(\text{congruence}) & \frac{\sigma \leq \tau}{\forall \alpha. \sigma \leq \forall \alpha. \tau}
\end{array}$$

Figure 1: Mitchell’s Subtyping Inference Rules.

Until quite recently, it had been an open problem whether the subtyping relation is decidable. Longo, Milsted, and Soloviev recently devised a new axiomatization of the subtyping relation which avoids having an explicit rule for transitivity [LMS95]. Using this new axiomatization, Tiurny and Urzyczyn recently proved the undecidability of the subtyping relation by a reduction from the halting problem for 2-counter automata [TU95]. Also using this new axiomatization, I devised a syntax-directed rule system for subtyping and proven its correctness. Then using the syntax-directed rules, I proved the subtyping relation to be undecidable by technique totally different from Tiurny and Urzyczyn’s and significantly simpler, namely a reduction from the problem of semi-unification [Wel95b]. The undecidability of subtyping implies the undecidability of type checking for $F+\eta$, because there is a trivial reduction from subtyping to type checking where the subtyping question “ $\sigma \leq \tau$ ” becomes the type checking question “ $\{x:\sigma\} \vdash x : \tau$ ”.

1.2 Contributions of This Paper It is an important question whether typability is decidable for $F+\eta$. The combination of System F with the (η) rule was considered as a type system long before Mitchell discovered the subtyping system to which it is equivalent. Many people have expressed interest in the question of decidability of typability but no solutions or partial solutions have been given.³ To a certain extent, the intensive research attempting to discover whether typability is decidable for System F took priority, since it is important to understand the base case before exploring the variations.

The main contribution of this paper is the first proof that typability is undecidable for $F+\eta$. The methods used in [Wel94] to reduce the problem of type checking to typability for System F are adapted to work for $F+\eta$. The top-level outline is very similar, but the fine details differ greatly. At the top level, the proof begins by showing there exists a typable λ -term J such that in every typing of J , its bound variable x is assigned the type $\alpha \rightarrow \alpha$. The term with this property is much more complicated for $F+\eta$ than for System F. Then, starting from the term J , contexts (terms with holes) are constructed which simulate more and more complex type assignments, forcing particular bound variables to be assigned exactly the desired types. The context constructions are identical to those used in the proof for System F up to the point where all contexts with universal types

³The author’s major advisor has mentioned this problem in grant proposals since at least 1988.

can be simulated. At that point, it is not clear how to simulate any arbitrary type assignment, but the portion of the type checking problem that can be handled turns out to be undecidable, thus proving the undecidability of typability for $F+\eta$.

We conjecture that the full problem of type checking can be reduced to typability in $F+\eta$. T. Jim has recently reduced typability in $F+\eta$ to subtyping using principal typings of terms in distinct operator form [Jim95]. Since typability is reducible to subtyping, and since subtyping is reducible to type checking, our conjecture would imply that all three problems are equivalent.

There are other systems related to System F and $F+\eta$ for which the decidability of typability is unknown. If a term like our term J can be constructed for a type system, then the overall outline of our methods can probably be applied, but the details will likely differ.

1.3 Acknowledgements Trevor Jim made me aware of Tiurny’s paper on bicoercibility [Tiu95], which was essential for my understanding of the subtyping relation. Assaf Kfoury provided vital support and encouragement.

2 Definitions and Foundation

This section introduces basic definitions, notation, and nomenclature for standard concepts, background results by other researchers, and some minor lemmas regarding these notions.

2.1 General Notation For any entity X , the notation \vec{X} denotes the sequence $X_1X_2\cdots X_n$ for some natural number n that is either unspecified or clear from the context. The sequence \vec{X} may stand for either the set $\{X_1, X_2, \dots, X_n\}$ or the comma-separated sequence X_1, X_2, \dots, X_n , depending on the context.

For any function fun , the notation $DOM(fun)$ denotes the *domain* of fun and $RAN(fun)$ denotes the *range* of fun . For any set S and function fun , the application $fun(S)$ denotes the set $\{fun(s) \mid s \in S\}$.

For any binary relation R , the notation “ R^* ” denotes the transitive, reflexive closure of R . The notation “ R^{-1} ” denotes the relation such that $R(x, y) \Leftrightarrow R^{-1}(y, x)$ for all x, y , while “ R^{+1} ” denotes R . If R is denoted by a directional symbol (such as “ $<$ ”), the reversed symbol (such as “ $>$ ”) denotes R^{-1} .

2.2 Terms Our notation for the λ -calculus generally follows Barendregt’s [Bar84]. The set of all λ -terms Λ is built from the countably infinite set of λ -term variables \mathcal{V} using application and abstraction as specified by this grammar:

$$\Lambda ::= \mathcal{V} \mid (\Lambda \Lambda) \mid (\lambda \mathcal{V}.\Lambda)$$

Small Roman letters from the beginning or end of the alphabet (e.g., a, b, c, x, y, z) are used as metavariables ranging over \mathcal{V} and capital Roman letters as metavariables ranging over Λ . When writing λ -terms, application associates to the left so that $MNP \equiv (MN)P$. The scope of “ $\lambda x.$ ” extends as far to the right as possible. The notation $\lambda \vec{x}.M$ stands for $\lambda x_1 \cdots \lambda x_k.M$ for some appropriate k . We assume at all times that every λ -term M obeys the restriction that no variable is λ -bound more than once and no variable occurs both λ -bound and free in M . The notation (**let** $v = P$ **in** Q) stands for $((\lambda v.Q)P)$.

As usual, $FV(M)$ and $BV(M)$ denote the free and λ -bound variables of a λ -term M . A λ -term is *open* if it has no λ -bound variables and is *closed* if it has no free variables. $N \subset M$ denotes that N is a proper subterm of M and $N \subseteq M$ includes the possibility that $N \equiv M$. When there

are n distinct occurrences of a particular subterm N in M , then the distinct occurrences will be distinguished by parenthesized superscripts which are numbered from left to right: $N^{(1)}, \dots, N^{(n)}$. The notation \square refers to some unspecified subterm whose actual value does not matter.

A *context* $C[\]$ is a λ -term with one or more holes. Contexts are specified by the grammar

$$\Lambda[\] ::= \mathcal{V} \mid [\] \mid (\Lambda[\] \Lambda[\]) \mid (\lambda \mathcal{V} . \Lambda[\])$$

If M is a λ -term and $C[\]$ is a context with one hole, then $C[M]$ denotes the result of inserting M into the hole in $C[\]$, *including* the capture of free variables in M by the λ -bindings of $C[\]$. A context with more than one hole is written $C[\ \dots, \]$, where $C[M_1, \dots, M_n]$ denotes inserting the terms M_1, \dots, M_n into the holes of $C[\ \dots, \]$, which are numbered from left to right. For a context $C[\]$ with one hole, define $\text{BHV}(C[\])$ to be the subset of λ -bound variables in $\text{BV}(C[\])$ whose scope includes the hole in $C[\]$.

The notion of α -conversion represents the fact that the names of bound variables are irrelevant. A term of the form $C[\lambda x.M]$ α -reduces in one step to $C[\lambda y.M[x:=y]]$ for any variable y . The equivalence relation induced by α -reduction is called α -conversion. If M and N are λ -terms, then $M \equiv N$ means that M and N are identical after allowing α -conversion. Two contexts $C_1[\]$ and $C_2[\]$ are equal if $C_1[M] \equiv C_2[M]$ for every term M . Thus, α -conversion may only be performed in contexts on the bound variables whose scope does not include the hole.

A *substitution* is a partial function from \mathcal{V} to Λ which is only defined for finitely many inputs. The notation $[x_1:=N_1, \dots, x_n:=N_n]$, which may be abbreviated as $[\vec{x}:=\vec{N}]$, denotes a substitution S such that $S(x_i) = N_i$ for $1 \leq i \leq n$ and which is undefined elsewhere. Following ordinary usage, the application of a nameless substitution $[\vec{x}:=\vec{N}]$ to its argument x is written with the substitution on the right, e.g., $x[\vec{x}:=\vec{N}]$, while a named substitution S is written on the left, e.g., $S(x)$. A substitution S is automatically extended into a function from Λ to Λ as follows:

$$S(M) = \begin{cases} (S(N))(S(P)) & \text{if } M = NP, \\ (\lambda y.S(N[x:=y])) & \text{if } M = (\lambda x.N), \text{ where } y \text{ is fresh,} \\ S(x) & \text{if } M = x \in \mathcal{V} \text{ and } S(x) \text{ defined,} \\ y & \text{if } M = y \in \mathcal{V} \text{ and } S(y) \text{ undefined.} \end{cases}$$

2.3 Types The set of types \mathbb{T} is built from the countably infinite set of type variables \mathbb{V} as specified by the grammar $\mathbb{T} ::= \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T}) \mid (\forall \mathbb{V} . \mathbb{T})$. Small Greek letters from the beginning of the alphabet (e.g., $\alpha, \beta, \gamma, \delta$) are metavariables over \mathbb{V} and small Greek letters towards the end of the alphabet (e.g., σ and τ) are metavariables over \mathbb{T} . Capital Roman letters in a “blackboard-bold” style (e.g., \mathbb{X} and \mathbb{Y}) are metavariables over subsets of \mathbb{T} . When writing types, the arrows associate to the right so that $\sigma \rightarrow \tau \rightarrow \rho$ stands for the type $\sigma \rightarrow (\tau \rightarrow \rho)$. The scope of “ \forall .” extends as far to the right as possible. The notation $\forall \vec{\alpha} . \sigma$ stands for $\forall \alpha_1 . \dots . \forall \alpha_k . \sigma$, which in turn stands for $\forall \alpha_1 . (\dots (\forall \alpha_k . \sigma) \dots)$. The symbol “ \perp ” is shorthand for $\forall \alpha . \alpha$.

The notion of α -conversion on types is defined similarly to α -conversion on λ -terms. Substitutions on types are defined similarly to substitutions on λ -terms. A *renaming* of free type variables is a substitution $[\vec{\alpha}:=\vec{\beta}]$ whose range contains only type variables.

The expressions $\text{FTV}(\tau)$ and $\text{BTV}(\tau)$ denote the free and \forall -bound type variables of type τ , respectively. For a set of types \mathbb{X} , the notation $\text{FTV}(\mathbb{X})$ denotes $\bigcup_{\tau \in \mathbb{X}} \text{FTV}(\tau)$. The notation $\forall . \sigma$ means $\forall \vec{\alpha} . \sigma$ where $\vec{\alpha} = \text{FTV}(\sigma)$. A type is *open* if it has no bound type variables and is *closed* if it has no free type variables.

We have several conventions about how quantifiers in types are treated.

1. Reordering of adjacent quantifiers is allowed at any time as well as α -conversion. For example, we consider the written type instances $\forall\alpha.\forall\beta.\alpha \rightarrow \beta$, $\forall\beta.\forall\alpha.\beta \rightarrow \alpha$, and $\forall\beta.\forall\alpha.\alpha \rightarrow \beta$ to all represent the same type.
2. Using α -conversion we freely assume that in any written type instance no variable is \forall -bound more than once, that the \forall -bound type variables of any two written type instances are disjoint, and that all \forall -bound type variables of any written type instance are disjoint from the free type variables of another written type instance.
3. If $\sigma = \forall\alpha.\tau$ and $\alpha \notin \text{FTV}(\tau)$, then “ $\forall\alpha$ ” is a *redundant* quantifier. We do not allow redundant quantifiers to affect the meaning of a type. For example, we consider the written type instances $\forall\beta.\forall\alpha.\alpha$ and $\forall\alpha.\alpha$ to represent the same types.

The notation $\sigma \subset \tau$ means that the type σ is properly *embedded* in the type τ . This relation is the transitive closure of the smallest relation such that $\sigma \subset \tau$ if there exist both $\vec{\alpha}$ and ρ such that $\tau = \forall\vec{\alpha}.\sigma \rightarrow \rho$ or $\tau = \forall\vec{\alpha}.\rho \rightarrow \sigma$. This definition is a bit unusual since $\forall\beta.\sigma \not\subset \forall\alpha.\forall\beta.\sigma$. The notation $\sigma \subseteq \tau$ includes the possibility that $\sigma = \tau$.

A *type context* $\sigma[\]$ is a type with one or more holes. Type contexts are specified by this grammar:

$$\mathbb{T}[\] ::= \mathbb{V} \mid [\] \mid (\mathbb{T}[\] \rightarrow \mathbb{T}[\]) \mid (\forall \mathbb{V}.\mathbb{T}[\])$$

If $\sigma[\]$ is a type context with one hole, then $\sigma[\tau]$ is the result of placing τ in the hole in $\sigma[\]$ *including* the possible capture of free type variables of τ by \forall -bindings in $\sigma[\]$. All of the conventions that apply to term contexts also apply to type contexts. When determining whether a \forall -binding is redundant, let $\text{FTV}([\]) = \mathbb{V}$.

2.3.1 Types as Trees

We may view a type as a binary tree where each “ \rightarrow ” corresponds to an internal node, type variable occurrences are leaf nodes, and each quantifier associates some tree node with some number of leaf nodes.

Let $\mathcal{P} = \{L, R\}^*$ be the set of finite paths in binary trees starting from the root where ε denotes the empty path. The meaning of L (respectively, R) is to go from a tree node to its *left* (respectively, *right*) child. Let $\mathcal{P}^\omega = \{L, R\}^\omega$ be the set of infinite paths. Capital Greek letters (e.g. Γ, Σ, Π) are metavariables ranging over \mathcal{P} and \mathcal{P}^ω . The path concatenation operation $\Pi \cdot \Sigma$ extends the finite path Π by the path Σ . The “ \cdot ” is usually omitted. The path prefix predicate “ \leq ” is defined so that $\Pi \leq \Sigma$ if and only if $\Sigma = \Pi\Delta$ for some path Δ . If $\Pi \leq \Sigma$ and $\Pi \neq \Sigma$, this is denoted $\Pi < \Sigma$. The left quotient (path prefix removal) operator “ \setminus ” is defined so that $\Pi \setminus \Sigma = \Delta$ if $\Sigma = \Pi\Delta$ and is undefined otherwise. The length (number of symbols) of a path Π is written $|\Pi|$. Given a path Π , its repetition k times is written as Π^k . The notation Π^ω denotes the repetition of Π infinitely many times. These predicates and operators work on sets of paths in the expected way, e.g., $\Pi \cdot \{\Sigma_1, \dots, \Sigma_n\} = \{\Pi\Sigma_1, \dots, \Pi\Sigma_n\}$ and $\Pi \setminus \{\Sigma_1, \dots, \Sigma_n\} = \{\Delta \mid \Sigma_i = \Pi\Delta \text{ for } 1 \leq i \leq n\}$.

A particular way to write type trees turns out to be very convenient. The *leaf-group set* representation of a type tree is as follows. A type σ is a set of leaf groups $\{G_1, \dots, G_n\}$ where for $1 \leq i \leq n$ it holds that G_i is a pair such that either $G_i = \langle \alpha_i, P_i \rangle$ or $G_i = \langle \Pi_i, P_i \rangle$ where $\emptyset \neq P_i \subset \mathcal{P}$. The sequence P_1, \dots, P_n is a partition of the leaves of σ . If the first item in the pair G_i is the variable α_i , this indicates that the free variable α_i occurs at each of the leaves in P_i . No two leaf groups mention the same free variable. If the first item is the path Π_i , then all of the leaves in P_i are quantified by the same quantifier which is located at Π_i .

It is often useful to say that a particular pattern of quantification occurs in a type without having to specify all of the leaves bound by the quantifier or the exact depth of the leaves. A leaf-group pattern is either a pair $\langle \alpha, P \rangle$ or a pair $\langle \Pi, P \rangle$ satisfying the following properties:

1. $P \subset \mathcal{P} \cup \mathcal{P}^\omega$, i.e., some of the members of P may be finite paths and some may be infinite.
2. If a path Π is the first item in the pair, then for every $\Sigma \in P$ it holds that $\Pi \leq \Sigma$, i.e., the path Π must be a legitimate position for a quantifier binding the path Σ .
3. For every $\Sigma, \Gamma \in P$ it holds that $\Sigma \not\leq \Gamma$, i.e., no path in P may be a prefix of another.

A leaf-group pattern $\langle X, P \rangle$ matches a type σ , written $\langle X, P \rangle \triangleleft \sigma$, if there is a pair $\langle X, P' \rangle \in \sigma$ such that for every finite member $\Pi \in P \cap \mathcal{P}$ it holds that Π matches some member of P' exactly, i.e., $\Pi \in P'$, and for every infinite member $\Pi \in P \cap \mathcal{P}^\omega$ there is some path $\Pi' \in P'$ which Π extends, i.e., $\Pi' < \Pi$.

Our type tree representation allows us to define a number of useful functions. The function *sign* defines whether a position in a type is *positive* or *negative*:

$$\text{sign}(\Pi) = \begin{cases} +1 & \text{if } \Pi \in \mathcal{P} \text{ has even number of } Ls, \\ -1 & \text{if } \Pi \in \mathcal{P} \text{ has odd number of } Ls, \\ \text{undefined} & \text{if } \Pi \in \mathcal{P}^\omega \end{cases}$$

The functions *leaf* and *leaves* return information about the shape of a type tree:

$$\begin{aligned} \text{leaf}(\sigma, \Pi) &= \begin{cases} \Sigma & \text{if } \Sigma \leq \Pi, \Sigma \in P, \text{ and } \langle X, P \rangle \in \sigma, \\ \text{undefined} & \text{otherwise.} \end{cases} \\ \text{leaves}(\sigma) &= \{ \Pi \mid \text{leaf}(\sigma, \Pi) = \Pi \} \end{aligned}$$

The following functions *quant* and *leafvar* are defined in disjoint situations. When a leaf is quantified, *quant* gives the position of the quantifier. Otherwise, *leafvar* gives the name of the free variable at the leaf.

$$\begin{aligned} \text{quant}(\sigma, \Pi) &= \begin{cases} \Delta & \text{if } \Sigma \leq \Pi, \Sigma \in P, \langle \Delta, P \rangle \in \sigma, \text{ and } \Delta \in \mathcal{P}, \\ \text{undefined} & \text{otherwise.} \end{cases} \\ \text{leafvar}(\sigma, \Pi) &= \begin{cases} \alpha & \text{if } \Sigma \leq \Pi, \Sigma \in P, \langle \alpha, P \rangle \in \sigma, \text{ and } \alpha \in \mathbb{V}, \\ \text{undefined} & \text{otherwise.} \end{cases} \end{aligned}$$

The function *quantsign* is directly inspired by Tiuryn's notion of positive and negative marking of leaves in types [Tiu95]:

$$\text{quantsign}(\sigma, \Pi) = \begin{cases} \text{sign}(\text{quant}(\sigma, \Pi)) & \text{if } \text{quant}(\sigma, \Pi) \text{ defined,} \\ 0 & \text{if } \text{quant}(\sigma, \Pi) \text{ not defined but } \text{leaf}(\sigma, \Pi) \text{ is,} \\ \text{undefined} & \text{otherwise.} \end{cases}$$

The functions *hole* and *holesign* give information about the hole in a type context:

$$\begin{aligned} \text{hole}(\sigma[\]) &= \Pi \text{ where } \langle \beta, \{\Pi\} \rangle \in \sigma[\beta] \text{ for fresh variable } \beta. \\ \text{holesign}(\sigma[\]) &= \text{sign}(\text{hole}(\sigma[\])) \end{aligned}$$

$$\begin{array}{lll}
(\text{sub}) & \forall \vec{\alpha}. \sigma \leq \forall \vec{\beta}. (\sigma[\vec{\alpha} := \vec{\tau}]) & \vec{\beta} \notin \text{FTV}(\forall \vec{\alpha}. \sigma) \\
(\text{distr}) & \forall \alpha. (\sigma \rightarrow \tau) \leq (\forall \alpha. \sigma) \rightarrow (\forall \alpha. \tau) & \\
(\rightarrow) & \frac{\sigma_2 \leq \sigma_1, \quad \tau_1 \leq \tau_2}{\sigma_1 \rightarrow \tau_1 \leq \sigma_2 \rightarrow \tau_2} & \\
(\text{trans}) & \frac{\rho \leq \sigma, \quad \sigma \leq \tau}{\rho \leq \tau} & \\
(\text{congruence}) & \frac{\sigma \leq \tau}{\forall \alpha. \sigma \leq \forall \alpha. \tau} &
\end{array}$$

Figure 2: Mitchell's Subtyping Inference Rules.

$$\begin{array}{lll}
(\text{ax}) & \sigma \vdash_{co} \sigma & \\
(\rightarrow) & \frac{\sigma_2 \vdash_{co} \sigma_1, \quad \tau_1 \vdash_{co} \tau_2}{\sigma_1 \rightarrow \tau_1 \vdash_{co} \sigma_2 \rightarrow \tau_2} & \\
(\forall\text{-left}) & \frac{\sigma[\alpha := \rho] \vdash_{co} \tau}{\forall \alpha. \sigma \vdash_{co} \tau} & \\
(\forall_n\text{-right}) & \frac{\sigma \vdash_{co} \rho_1 \rightarrow \cdots \rightarrow \rho_n \rightarrow \tau}{\sigma \vdash_{co} \rho_1 \rightarrow \cdots \rightarrow \rho_n \rightarrow (\forall \alpha. \tau)} & \alpha \notin \text{FTV}(\{\sigma, \vec{\rho}\})
\end{array}$$

Figure 3: System F_{co}^+ Subtyping Inference Rules.

The following functions *height* and *parheight* give information about the size of a type. The function *height* simply measures the tree's height while *parheight* measures the heights of the parameter types within a type.

$$\begin{aligned}
\text{height}(\sigma) &= \max\{n \mid \Pi \in \text{leaves}(\sigma), |\Pi| = n - 1\} \\
\text{parheight}(\sigma) &= \max(\{0\} \cup \{n \mid R^k L\Pi \in \text{leaves}(\sigma), |L\Pi| = n\})
\end{aligned}$$

2.4 Subtyping A type σ is a *subtype* of type τ if and only if there is a derivation that $\sigma \leq \tau$ in Mitchell's system in Figure 2.

Longo, Milsted, and Soloviev give another axiomatization of the subtyping relation called System F_{co}^+ ⁴, which is presented in Figure 3 [LMS95]. This rule system may be used interchangeably with Mitchell's.

⁴This name is used because when the inference rules of the system are labelled with proof terms, it becomes a fragment of System F. The *co* subscript stands for *coercion*.

Theorem 2.1 For all types σ and τ , it holds that $\sigma \leq \tau$ if and only if $\sigma \vdash_{co} \tau$.

Proof: This is Theorem 11 in [LMS95]. ■

$$\begin{array}{l}
(\text{var}) \quad \alpha \lesssim \alpha \\
(\perp) \quad \perp \lesssim \sigma \\
(\Rightarrow) \quad \frac{\tau_L \lesssim \forall \vec{\beta}.(\sigma_L[\vec{\alpha}:=\vec{\rho}]), \quad \forall \vec{\beta}.(\sigma_R[\vec{\alpha}:=\vec{\rho}]) \lesssim \tau_R}{\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \lesssim \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)} \\
\text{where } \vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R) \text{ and } \vec{\gamma} \notin \text{FTV}(\forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R))
\end{array}$$

Figure 4: Syntax-Directed Subtyping Inference Rules.

Consider the rule system in Figure 4, which we call the *syntax-directed* subtyping rules. This system is syntax-directed because if $\sigma \lesssim \tau$ can be derived, then the last rule used in the derivation is uniquely determined by the syntax of σ . The syntax-directed nature of this rule system gives us the following nice property.

Lemma 2.2 If $\sigma \lesssim \tau$ and $\sigma = \forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R)$ for some types σ_L and σ_R and some type variables $\vec{\alpha}$, then $\tau = \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$ for some types τ_L and τ_R and some type variables $\vec{\gamma} \notin \text{FTV}(\sigma)$ and there exist some types $\vec{\rho}$ and some type variables $\vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R)$ such that:

$$\tau_L \lesssim \forall \vec{\beta}.(\sigma_L[\vec{\alpha}:=\vec{\rho}]) \quad \text{and} \quad \forall \vec{\beta}.(\sigma_R[\vec{\alpha}:=\vec{\rho}]) \lesssim \tau_R$$

Proof: See [Wel95b]. ■

This rule system is yet another axiomatization of the subtyping relation.

Theorem 2.3 For all types σ and τ , it holds that $\sigma \leq \tau$ if and only if $\sigma \lesssim \tau$.

Proof: See [Wel95b]. ■

Tiurnyn has devised a set of rewriting-style rules for the subtyping relation [Tiu95]. We use a modified version of these rules shown in Figure 5. Although the presentation here is not identical to that in [Tiu95], they are equivalent. Tiurnyn’s system has rules for introducing and removing redundant quantifiers, which we omit because this need is handled by our definition of type contexts. Recall that the notation “ \sqsubset^* ” denotes the transitive, reflexive closure of “ \sqsubset ”. Since the rules reverse directions at positive and negative positions, we use the notation “ \sqsubset^{-1} ” to mean “ \sqsupset ”.

It is possible for $\rho \sqsubset \rho$ by rule (h-inst s). For example, if $\tau[\] = \forall \beta.[\]$ and $\beta \notin \text{FTV}(\forall \alpha.\sigma)$, then $\tau[\forall \alpha.\sigma] \sqsubset \tau[\sigma[\alpha:=\beta]]$, where both types are the same. Such uses of (h-inst s) are called *redundant* and we assume that redundant uses do not occur.

$$\text{(h-inst } s) \quad \tau[\forall\alpha.\sigma] \sqsubset^s \tau[\sigma[\alpha:=\rho]] \quad s = \text{holesign}(\tau[\])$$

$$\text{(h-distr } s) \quad \tau[\forall\alpha.(\sigma \rightarrow \rho)] \sqsubset^s \tau[(\forall\alpha.\sigma) \rightarrow (\forall\alpha.\rho)] \quad s = \text{holesign}(\tau[\])$$

Figure 5: Rewriting-Style Subtyping Inference Rules.

The rule (h-distr s) is more powerful than it needs to be. Consider the following restriction of (h-distr s):

$$\tau[\forall\alpha.(\sigma \rightarrow \rho)] \sqsubset^s \tau[\sigma \rightarrow (\forall\alpha.\rho)] \quad \text{(h-shift } s)$$

The rule (h-distr s) can be replaced by (h-shift s) without losing any power. This restriction is inspired by the restriction in rule (A3) of the bicoercibility rules by Tiurnyn in Figure 6.

Lemma 2.4 *If $\sigma \sqsubset^s \tau$ by rule (h-distr s), then $\sigma \sqsubset^{s,*} \tau$ using only rules (h-inst s) and (h-shift s).*

Proof: Consider the following rule, which lifts the restriction “ α occurs negatively in ρ ” from (h-shift s):

$$\tau[\forall\alpha.(\sigma \rightarrow \rho)] \sqsubset^s \tau[\sigma \rightarrow (\forall\alpha.\rho)] \quad s = \text{holesign}(\tau[\]), \alpha \notin \text{FTV}(\sigma) \quad \text{(h-shift' } s)$$

We prove that (h-shift' s) is admissible using the rules (h-inst s) and (h-shift s). Assuming the premises of (h-shift' s), which are that $s = \text{holesign}(\tau[\])$ and $\alpha \notin \text{FTV}(\sigma)$, the desired result will be shown. If α occurs negatively in ρ , then the desired result follows immediately from rule (h-shift s). If α occurs only positively in ρ , then $\tau[\forall\alpha.(\sigma \rightarrow \rho)] \sqsubset^s \tau[S(\sigma \rightarrow \rho)] = \tau[\sigma \rightarrow S(\rho)]$ by (h-inst s) where S is the substitution $[\alpha:=\perp]$. Now we show that $\tau[\sigma \rightarrow S(\rho)] \sqsubset^{s,*} \tau[\sigma \rightarrow (\forall\alpha.\rho)]$ using only rule (h-inst s). For each occurrence of α in ρ , we will use the rule (h-inst s) to replace the corresponding occurrence of \perp in $\rho[\alpha:=\perp]$ by α , using the type context to capture it at the right spot. Let $\langle\alpha, P\rangle \in \rho$ and pick some arbitrary $\Pi \in P$. Since $\text{sign}(\Pi) = +1$, it is clear that $\rho[\alpha:=\perp] = \pi[\perp]$ for some type context $\pi[\]$ such that $\text{hole}(\pi[\]) = \Pi$ and $\text{holesign}(\pi[\]) = +1$. Let $\theta[\] = \tau[\sigma \rightarrow (\forall\alpha.\pi[\])]$. It holds that $\text{holesign}(\theta[\]) = s$. Thus,

$$\tau[\sigma \rightarrow S(\rho)] = \theta[\perp] \sqsubset^s \theta[\alpha]$$

by rule (h-inst s). Repeating this process for each occurrence of α in ρ produces the desired result.

The rest of the proof is to show that (h-distr s) is admissible using rules (h-inst s) and (h-shift' s). Assume the premises of (h-distr s), which are merely that $s = \text{holesign}(\tau[\])$. By rule (h-inst $-s$) it holds that $\tau[\forall\alpha.(\sigma \rightarrow \rho)] \sqsubset^s \tau[\forall\alpha.((\forall\alpha.\sigma) \rightarrow \rho)]$. Then (h-shift' s) proves that $\tau[\forall\alpha.((\forall\alpha.\sigma) \rightarrow \rho)] \sqsubset^s \tau[(\forall\alpha.\sigma) \rightarrow (\forall\alpha.\rho)]$, which is the desired result. ■

Unfortunately, Tiurnyn does not provide a proof of the correctness of his rules in [Tiu95], so we supply the following lemmas and theorem to handle this need.

Lemma 2.5 *If $\sigma \leq \rho$, then $\tau[\sigma] \leq^s \tau[\rho]$ where $s = \text{holesign}(\tau[\])$.*

Proof: Let $\Pi = \text{hole}(\tau[\])$. By induction on the length of Π .

1. ($\Pi = \varepsilon$) Then $\tau[\] = \forall \vec{\alpha}.[\]$ for some sequence of variables $\vec{\alpha}$. By (congruence), $\forall \vec{\alpha}.\sigma \leq \forall \vec{\alpha}.\rho$, which is exactly the desired result $\tau[\sigma] \leq^s \tau[\rho]$.
2. ($\Pi > \varepsilon$) Assume $\Pi = L\Pi'$ for some Π' . (The reasoning is almost identical when $\Pi = R\Pi'$.) Then $\tau[\] = \forall \vec{\alpha}.\tau_L[\] \rightarrow \tau_R$ for some type context $\tau_L[\]$ and some type τ . It holds that $\text{holesign}(\tau[\]) = -\text{holesign}(\tau_L[\])$. By induction, $\tau_L[\sigma] \leq^{-s} \tau_L[\rho]$. By (\rightarrow), $\tau_L[\sigma] \rightarrow \tau_R \leq^s \tau_L[\rho] \rightarrow \tau_R$. By (congruence), $\forall \vec{\alpha}.\tau_L[\sigma] \rightarrow \tau_R \leq^s \forall \vec{\alpha}.\tau_L[\rho] \rightarrow \tau_R$, which is exactly the desired result $\tau[\sigma] \leq^s \tau[\rho]$.

■

Lemma 2.6 *If $\sigma \sqsubset^s \rho$, then $\tau[\sigma] \leq^{st} \tau[\rho]$ where $t = \text{holesign}(\tau[\])$.*

Proof: It must be the case that $\sigma = \pi[\sigma'] \sqsubset^s \pi[\rho'] = \rho$ by rule R where $s = \text{holesign}(\pi[\])$ and one of these four possibilities holds:

$$\begin{array}{lll}
R \text{ is (h-inst +1),} & \sigma' = \forall \alpha.\varphi, & \rho' = \varphi[\alpha := \theta] \\
R \text{ is (h-inst -1),} & \sigma' = \varphi[\alpha := \theta], & \rho' = \forall \alpha.\varphi \\
R \text{ is (h-distr +1),} & \sigma' = \forall \alpha.(\varphi_L \rightarrow \varphi_R), & \rho' = (\forall \alpha.\varphi_L) \rightarrow (\forall \alpha.\varphi_R) \\
R \text{ is (h-distr -1),} & \sigma' = (\forall \alpha.\varphi_L) \rightarrow (\forall \alpha.\varphi_R), & \rho' = \forall \alpha.(\varphi_L \rightarrow \varphi_R)
\end{array}$$

In any of these cases, it is an immediate result that $\tau[\pi[\sigma']] \sqsubset^{st} \tau[\pi[\rho']]$. ■

Theorem 2.7 *For all types σ and τ , it holds that $\sigma \leq \tau$ if and only if $\sigma \sqsubset^* \tau$.*

Proof: Each direction is proved separately.

1. (\Leftarrow) Since “ \leq ” is transitive by rule (trans) and reflexive by rule (sub), it is sufficient to show that each of the rules (h-inst s) and (h-distr s) are admissible for “ \leq ”.
 - (a) (h-inst s) By (sub), $\forall \alpha.\sigma \leq \sigma[\alpha := \rho]$. By Lemma 2.5, $\tau[\forall \alpha.\sigma] \leq^s \tau[\sigma[\alpha := \rho]]$ where $s = \text{holesign}(\tau)$.
 - (b) (h-distr s) By (distr), $\forall \alpha.(\sigma \rightarrow \tau) \leq (\forall \alpha.\sigma) \rightarrow (\forall \alpha.\tau)$. By Lemma 2.5, it holds that $\tau[\forall \alpha.\sigma] \leq^s \tau[\sigma[\alpha := \rho]]$ where $s = \text{holesign}(\tau)$.
2. (\Rightarrow) Each of the rules for “ \leq ” are admissible for “ \sqsubset^* ”.
 - (a) (sub) It is desired to show that $\forall \vec{\alpha}^n.\sigma \sqsubset^* \forall \vec{\beta}.(\sigma[\vec{\alpha}^n := \vec{\rho}^n])$ where $\vec{\beta} \notin \text{FTV}(\forall \vec{\alpha}^n.\sigma)$. By α -conversion, assume that $\vec{\alpha} \notin \text{FTV}(\vec{\rho})$. Let i range over $\{0, \dots, n\}$. Let $\pi_i[\] = \forall \vec{\beta}.\forall \alpha_n. \dots \forall \alpha_{i+1}.[\]$ and let S_i be the substitution $[\vec{\alpha}^i := \vec{\rho}^i]$. If $\vec{\beta} \notin \text{FTV}(\forall \vec{\alpha}^n.\sigma)$, then $\forall \vec{\alpha}.\sigma = \pi_0[S_0(\sigma)]$. By (h-inst +1) when $i > 0$ it holds that $\pi_i[\forall \alpha_i.S_{i-1}(\sigma)] \sqsubset \pi_i[S_i(\sigma)]$. It is the case when $i < n$ that $\pi_i[S_i(\sigma)] = \pi_{i+1}[\forall \alpha_{i+1}.S_i(\sigma)]$. Thus, when $i > 0$, it holds that $\pi_{i-1}[S_{i-1}(\sigma)] \sqsubset \pi_i[S_i(\sigma)]$. Thus, $\forall \vec{\alpha}.\sigma = \pi_0[S_0(\sigma)] \sqsubset^* \pi_n[S_n(\sigma)] = \forall \vec{\beta}.(\sigma[\vec{\alpha} := \vec{\rho}])$.
 - (b) (distr) This rule is a special case of (h-distr +1).
 - (c) (\rightarrow) It is given that $\tau_L \sqsubset^* \sigma_L$ and $\sigma_R \sqsubset^* \tau_R$ and it is desired to show that $\sigma_L \rightarrow \sigma_R \sqsubset^* \tau_L \rightarrow \tau_R$. Let $\pi_1[\] = [\] \rightarrow \sigma_R$ and $\pi_2[\] = \tau_L \rightarrow [\]$. By Lemma 2.6,

$$\begin{array}{ll}
\pi_1[\tau_L] & \sqsubset^{-1,*} \pi_1[\sigma_L] \\
\pi_2[\sigma_R] & \sqsubset^{+1,*} \pi_2[\tau_R]
\end{array}$$

(A1)	$\sigma \leq \sigma$	
(A2)	$\forall \alpha. \forall \beta. \sigma \leq \forall \beta. \forall \alpha. \sigma$	
(A3)	$\forall \alpha. \sigma \leq \sigma[\alpha := \perp]$	all occurrences of α in σ are positive
(A4)	$\forall \alpha. (\sigma \rightarrow \tau) \leq \sigma \rightarrow \forall \alpha. \tau$	$\alpha \notin \text{FTV}(\sigma)$ and α occurs negatively in τ
(arrow)	$\frac{\sigma \leq \sigma', \quad \tau \leq \tau'}{\sigma \rightarrow \tau \leq \sigma' \rightarrow \tau'}$	(trans) $\frac{\sigma \leq \rho, \quad \rho \leq \tau}{\sigma \leq \tau}$
(quant)	$\frac{\sigma \leq \sigma'}{\forall \alpha. \sigma \leq \forall \alpha. \sigma'}$	(symm) $\frac{\sigma \leq \tau}{\tau \leq \sigma}$

Figure 6: Bicoercibility Inference Rules.

which is the same as

$$\begin{array}{l} \sigma_L \rightarrow \sigma_R \quad \sqsubset^* \quad \tau_L \rightarrow \sigma_R \\ \tau_L \rightarrow \sigma_R \quad \sqsubset^* \quad \tau_L \rightarrow \tau_R \end{array}$$

which is the desired result by transitivity.

(d) (trans) “ \sqsubset^* ” is transitive by definition.

(e) (congruence) It is given that $\sigma \sqsubset^* \tau$ and it is desired to show that $\forall \alpha. \sigma \sqsubset^* \forall \alpha. \tau$. Let $\pi[\] = \forall \alpha. [\]$. By Lemma 2.6, $\pi[\sigma] \sqsubset^* \pi[\tau]$, which is exactly the desired result.

■

Definition 2.8 The subtyping problem: Given an arbitrarily chosen pair of types σ and τ , is it the case that $\sigma \leq \tau$?

2.4.1 Bicoercibility

If σ is a subtype of τ and τ is also a subtype of σ , then σ and τ are *bicoercible*, which we write as $\sigma \leq \tau$. Tiurnyn devised the axiomatization in Figure 6 and proved that it captures precisely the notion of bicoercibility [Tiu95]. Tiurnyn used this rule system to prove the decidability of bicoercibility. The following lemma is an immediate consequence of Tiurnyn’s rule system.

Lemma 2.9 $\sigma \leq \tau$ if and only if all of the following properties hold:

1. $\text{leaves}(\sigma) = \text{leaves}(\tau)$, i.e., σ and τ have the same underlying tree skeleton.
2. For $\Pi \in \mathcal{P}^\omega$, $\text{quantsign}(\sigma, \Pi) = \text{quantsign}(\tau, \Pi)$, i.e., a leaf is quantified in σ if and only if it is quantified in τ and the quantifiers must have the same sign.
3. For $\Pi \in \mathcal{P}^\omega$, if $\text{leafvar}(\sigma, \Pi)$ and $\text{leafvar}(\tau, \Pi)$ are defined, then $\text{leafvar}(\sigma, \Pi) = \text{leafvar}(\tau, \Pi)$, i.e., the free type variable occurrences in σ match those in τ .

4. If $\Pi \in \mathcal{P}$ does not end with R , $\Delta \in \mathcal{P}$, $\Pi < \Delta$, $\text{sign}(\Pi) = -\text{sign}(\Delta)$, and $\Delta \in P \subset \mathcal{P}$, then there exists a number j such that $\langle \Pi R^j, P \rangle \triangleleft \sigma$ if and only if there exists k such that $\langle \Pi R^k, P \rangle \triangleleft \tau$. In other words, quantified leaf groups can not be split unless all leaves have the same sign as the quantifier.

Proof: Properties 1, 2, and 3 are proven by inductions on the structure of derivations which use the rules in Figure 6. Property 4 is also proven by induction by observing that the only rule that can split a leaf group is (A3), which can only be used when all of the quantified leaves have the same sign as the quantifier. ■

The following lemma shows that the (arrow) rule can be used in both directions.

Lemma 2.10 *If $(\sigma_L \rightarrow \sigma_R) \lesssim (\tau_L \rightarrow \tau_R)$ then $\sigma_L \lesssim \tau_L$ and $\sigma_R \lesssim \tau_R$.*

Proof: This is Lemma 6 in [Tiu95]. ■

2.5 Type Inference A pair $x:\sigma$ where $x \in \mathcal{V}$ and $\sigma \in \mathbb{T}$ is called a *type assumption*.⁵ A finite set of type assumptions $A = \{x_1:\sigma_1, \dots, x_n:\sigma_n\}$ which associates at most one type σ with any variable x is a *type assignment*.⁶ (Since we assume that no variable is λ -bound twice in a λ -term, viewing a type assignment as a set causes no problems.) A type assignment can be used as a function so that $A(x) = \sigma$ if $(x:\sigma) \in A$. The expression $\text{FTV}(A)$ denotes $\text{FTV}(\text{RAN}(A))$. An expression $A \vdash M : \tau$, where A is a type assignment, M is a λ -term, and $\tau \in \mathbb{T}$, is a *sequent*.⁷ We assume that throughout a sequent it is the case that all \forall -bound type variables are named distinctly from each other and that the \forall -bound and free type variables do not overlap (satisfied by α -conversion). A *derivation* is a finite sequence of sequents where each sequent is obtained from the preceding sequents according to the inference rules of the particular type system.

Let \mathcal{D} be a derivation $\tilde{\Delta}^m$ where for $1 \leq i \leq m$ each Δ_i is $A_i \vdash M_i : \tau_i$. The *global type assignment* of \mathcal{D} is $\mathcal{G}(\mathcal{D}) = \bigcup_{1 \leq i \leq m} A_i$. (Due to our assumption that no variable is λ -bound more than once in a term, $\mathcal{G}(\mathcal{D})$ is a well defined type assignment.) The *final derived type* (respectively, *initial derived type*) of a subterm occurrence N in \mathcal{D} is $\text{FDT}(\mathcal{D}, N) = \tau_i$ (resp., $\text{IDT}(\mathcal{D}, N)$) where Δ_i is the last (resp., first) sequent to mention N . When the derivation \mathcal{D} is clear from the context, $\mathcal{G}(x) = (\mathcal{G}(\mathcal{D}))(x)$, $\text{FDT}(N) = \text{FDT}(\mathcal{D}, N)$, and $\text{IDT}(N) = \text{IDT}(\mathcal{D}, N)$. If N is a subterm occurrence in a derivation \mathcal{D} , we will frequently want to make statements like the following:

$$\text{quantsign}(\text{IDT}(N), \Pi) = s \quad \text{and} \quad \text{quantsign}(\text{FDT}(N), \Pi) = s$$

In this case, the statement “ $\text{quantsign}(N, \Pi) = s$ ” is taken to be an abbreviation for the statement about $\text{IDT}(N)$ and $\text{FDT}(N)$. Statements using the functions *leaf*, *leaves*, *quant*, and *leafvar* may be abbreviated similarly.

A *typing* of the term M is a derivation whose last sequent is $A \vdash M : \tau$ for some type assignment A and type τ . A λ -term M is *typable* if and only if there is a typing for M .

Definition 2.11 The typability problem: Given an arbitrarily chosen λ -term M , is M typable?

Definition 2.12 The type-checking problem: Given arbitrarily chosen λ -term M , type assignment A , and type $\tau \in \mathbb{T}$, is there a typing of M that ends with the sequent $A \vdash M : \tau$?

VAR	$A \vdash x : A(x)$	
APP	$\frac{A \vdash M : \sigma \rightarrow \tau, \quad A \vdash N : \sigma}{A \vdash (M N) : \tau}$	
ABS	$\frac{A \cup \{x:\sigma\} \vdash M : \tau}{A \vdash (\lambda x.M) : \sigma \rightarrow \tau}$	
INST	$\frac{A \vdash M : \forall \alpha. \sigma}{A \vdash M : \sigma[\alpha := \tau]}$	
GEN	$\frac{A \vdash M : \sigma}{A \vdash M : \forall \alpha. \sigma}$	$\alpha \notin \text{FTV}(A)$

Figure 7: Type Inference Rules of System F.

2.5.1 System F and F Plus Eta

The usual type inference rules of System F are given in Figure 7. The VAR rule is the only axiom and handles variable occurrences. The APP rule handles application subterms while the ABS rule handles abstraction subterms. The name INST refers to *instantiation*, the process of removing a \forall -binding and replacing all of the bound occurrences with some type. The name GEN refers to *generalization*.

The type system $F+\eta$ is normally defined by adding to the rules of System F the (η) rule:

$$\frac{A \vdash \lambda x.Mx : \sigma \rightarrow \tau, \quad x \notin \text{FV}(M)}{A \vdash M : \sigma \rightarrow \tau} \quad (\eta)$$

However, Mitchell proved that this is equivalent to adding the (subsum) rule using the subtypings generated by the subtyping rules in Figure 2.

$$\frac{A \vdash M : \sigma}{A \vdash M : \tau} \quad \sigma \leq \tau \quad (\text{subsum})$$

Theorem 2.13 *The sequent $A \vdash M : \tau$ is derivable in System F extended with the (η) rule if and only if it is derivable in System F extended with the (subsum) rule.*

Proof: By Theorems 4 and 5 in [Mit90] (Theorems 13 and 18 in [Mit88]). ■

When working with $F+\eta$, it is convenient to remove the INST rule from the system and to require that the GEN rule be used only immediately after the ABS rule. Mitchell showed this to be possible because all uses of INST and most uses of GEN can be handled by (subsum). The type inference rules in Figure 8 reflect this change. Because they are more convenient, throughout this document we take them to be the definition of $F+\eta$.

⁵A type assumption has also been called a *declaration*, a *type assignment*, or a *type statement*.

⁶A type assignment has also been called a *basis*, an *environment*, or a *context*, although the term *context* usually indicates it is ordered.

⁷A sequent has also been called an *assertion*, a *type assignment formula*, a *judgement*, or a *typing*.

VAR	$A \vdash x : A(x)$	
ABS/GEN	$\frac{A \cup \{x:\sigma\} \vdash M : \tau}{A \vdash (\lambda x.M) : \forall \vec{\alpha}.(\sigma \rightarrow \tau)}$	$\vec{\alpha} \notin \text{FTV}(A)$
APP	$\frac{A \vdash M : \sigma \rightarrow \tau, \quad A \vdash N : \sigma}{A \vdash (M N) : \tau}$	
(subsum)	$\frac{A \vdash M : \sigma}{A \vdash M : \tau}$	$\sigma \leq \tau$

The definition of $\sigma \leq \tau$ is in Figure 2.

Figure 8: Type Inference Rules of $F+\eta$.

Theorem 2.14 *The sequent $A \vdash M : \tau$ is derivable in System F extended with the (subsum) rule if and only if it is derivable using the rules of Figure 8.*

Proof: Also by Theorems 4 and 5 in [Mit90] (Theorems 13 and 18 in [Mit88]). The only difference is that Mitchell formulates the APP rule as follows:

$$\frac{A \vdash M : \forall \vec{\alpha}.(\sigma \rightarrow \tau), \quad A \vdash N : \forall \vec{\alpha}.\sigma}{A \vdash (M N) : \forall \vec{\alpha}.\tau} \quad (\rightarrow E)$$

Mitchell's $(\rightarrow E)$ rule can be derived from our APP rule and the (subsum) rule using only the (distr) subtyping rule. ■

3 Quantifier Sign, Leaf Length, and Leaf Groups

This section proves some additional properties of subtyping which are used in Section 5. The most important lemma is based on the *quantsign* function. The main message of the lemma is that as subtyping proceeds, the *quantsign* value for any infinite path can only get smaller. This was first demonstrated by Tiurnyn [Tiu95]. The lemma also states useful properties of path lengths in types and how subtyping affects them.

Lemma 3.1 *If $\sigma \leq \tau$ then for all $\Pi \in \mathcal{P}^\omega$ the following properties hold:*

1. *If $\text{quantsign}(\sigma, \Pi) = 0 = \text{quantsign}(\tau, \Pi)$, then $\text{leaf}(\sigma, \Pi) = \text{leaf}(\tau, \Pi)$ and $\text{leafvar}(\sigma, \Pi) = \text{leafvar}(\tau, \Pi)$.*
2. *$\text{quantsign}(\sigma, \Pi) \geq \text{quantsign}(\tau, \Pi)$.*
3. *If $\text{leaf}(\sigma, \Pi) < \text{leaf}(\tau, \Pi)$, then $\text{quantsign}(\sigma, \Pi) = +1$.*
4. *If $\text{leaf}(\sigma, \Pi) > \text{leaf}(\tau, \Pi)$, then $\text{quantsign}(\tau, \Pi) = -1$.*

Proof: Each property is proven by induction on the structure of the derivation of $\sigma \leq \tau$ using the syntax-directed rules. For all four properties, the base cases of the rules (var) and (\perp) are trivial, so they are omitted. What remains is to prove the induction case of rule (\rightsquigarrow). The proof of property 2 depends on property 1. Properties 3 and 4 are proven by a mutual induction which depends on properties 1 and 2.

For all four properties, consider the case where $\sigma = \forall \vec{\alpha}.(\sigma_L \rightarrow \sigma_R)$ and $\tau = \forall \vec{\gamma}.(\tau_L \rightarrow \tau_R)$. By Lemma 2.2 there exist types $\vec{\rho}$ and type variables $\vec{\beta} \notin \text{FTV}(\sigma_L \rightarrow \sigma_R)$ such that $\tau_L \leq \hat{\sigma}_L$ and $\hat{\sigma}_R \leq \tau_R$ where $\hat{\sigma}_\Delta = \forall \vec{\beta}.(\sigma_\Delta[\vec{\alpha} := \vec{\rho}])$ for $\Delta \in \{L, R\}$. Define Π' and Δ so that $\Pi = \Delta \Pi'$ where $\Delta \in \{L, R\}$. Let $s = \text{sign}(\Delta)$.

1. It is given that $\text{quantsign}(\sigma, \Pi) = 0 = \text{quantsign}(\tau, \Pi)$. It is desired to show that $\text{leaf}(\sigma, \Pi) = \text{leaf}(\tau, \Pi)$ and $\text{leafvar}(\sigma, \Pi) = \text{leafvar}(\tau, \Pi)$. Let $\alpha_\rho = \text{leafvar}(\rho, \Pi)$ for $\rho \in \{\sigma, \tau\}$. $\alpha_\sigma \notin \vec{\beta}$ since $\alpha_\sigma \in \text{FTV}(\sigma)$. Thus, $\text{leafvar}(\hat{\sigma}_\Delta, \Pi') = \alpha_\sigma$. It is the case that $\text{leafvar}(\tau_\Delta, \Pi') = \alpha_\tau$. By induction on $\hat{\sigma}_\Delta \leq^s \tau_\Delta$, $\text{leaf}(\hat{\sigma}_\Delta, \Pi') = \text{leaf}(\tau_\Delta, \Pi')$ and $\alpha_\sigma = \alpha_\tau$. Thus, $\text{leaf}(\sigma, \Pi) = \Delta \cdot \text{leaf}(\hat{\sigma}_\Delta, \Pi') = \Delta \cdot \text{leaf}(\tau_\Delta, \Pi') = \text{leaf}(\tau, \Pi)$.
2. It is desired to show that $\text{quantsign}(\sigma, \Pi) \geq \text{quantsign}(\tau, \Pi)$. By cases on the value $n = \text{quantsign}(\sigma, \Pi)$.
 - (a) ($n = +1$) Immediate.
 - (b) ($n = 0$) Let $\alpha = \text{leafvar}(\sigma, \Pi)$. Observe $\alpha \in \text{FTV}(\sigma)$. Thus, $\alpha = \text{leafvar}(\hat{\sigma}_\Delta, \Pi')$. By induction, $0 = \text{quantsign}(\hat{\sigma}_\Delta, \Pi') \geq^s \text{quantsign}(\tau_\Delta, \Pi')$. Suppose it were the case that $\text{quantsign}(\tau_\Delta, \Pi') = 0$. Then, by property 1, $\text{leafvar}(\tau_\Delta, \Pi') = \alpha$ and since $\alpha \in \text{FTV}(\sigma)$ it must hold that $\alpha = \text{leafvar}(\tau, \Pi)$, so $\text{quantsign}(\tau, \Pi) = 0$. Suppose instead that $\text{quantsign}(\tau_\Delta, \Pi') = -s$. Then $\text{quantsign}(\tau, \Pi) = -1$.
 - (c) ($n = -1$) Then $\text{quantsign}(\hat{\sigma}_\Delta, \Pi') = -s$. By induction, $\text{quantsign}(\hat{\sigma}_\Delta, \Pi') \geq^s \text{quantsign}(\tau_\Delta, \Pi')$. Thus, $\text{quantsign}(\tau_\Delta, \Pi') = -s$ implying $\text{quantsign}(\tau, \Pi) = -1$.
3. It is given that $\text{leaf}(\sigma, \Pi) < \text{leaf}(\tau, \Pi)$. It is desired to show that $\text{quantsign}(\sigma, \Pi) = +1$. Suppose $\text{quantsign}(\sigma, \Pi) \neq +1$ and the following will show that a contradiction results. This assumption implies that $\text{leaf}(\sigma_\Delta, \Pi') = \text{leaf}(\hat{\sigma}_\Delta, \Pi') < \text{leaf}(\tau_\Delta, \Pi')$ and also $\text{quantsign}(\hat{\sigma}_\Delta, \Pi') \neq s$. If $\Delta = L$, then, by induction using property 4, $\text{quantsign}(\hat{\sigma}_L, \Pi') = -1 = s$, a contradiction. If instead $\Delta = R$, then, by induction using property 3, $\text{quantsign}(\hat{\sigma}_R, \Pi') = +1 = s$, a contradiction.
4. It is given that $\text{leaf}(\sigma, \Pi) > \text{leaf}(\tau, \Pi)$. It is desired to show that $\text{quantsign}(\tau, \Pi) = -1$. It is easy to see that $\text{leaf}(\hat{\sigma}_\Delta, \Pi') \geq \text{leaf}(\sigma_\Delta, \Pi') > \text{leaf}(\tau_\Delta, \Pi')$. If $\Delta = L$, then, by induction using property 3, $\text{quantsign}(\tau_L, \Pi') = +1 = -s$. If instead $\Delta = R$, then, by induction using property 4, $\text{quantsign}(\tau_R, \Pi') = -1 = -s$. Thus, $\text{quantsign}(\tau, \Pi) = -1$.

■

The second most important lemma states that under certain circumstances, leaves that belong to the same leaf group can not be split up as subtyping proceeds. The most important criteria to prevent the possibility of splitting is that one of the leaves must be at a positive position and one at a negative position. It is also necessary that the *quantsign* values of all of the paths can not be negative (or positive if using the lemma in the reverse direction). This lemma is proven using the rewriting-style subtyping rules. Recall that the notation “ \leq^{-1} ” means “ \geq ”.

Lemma 3.2 *If there exist types σ and τ , paths $\Pi, \Sigma_1, \dots, \Sigma_g \in \mathcal{P}$ where $g \geq 2$, an infinite path $\Gamma \in \mathcal{P}^\omega$, and a number f such that the following requirements are met:*

1. $\langle \Pi, \Pi R^f \cdot \{\Sigma_1, \dots, \Sigma_g\} \rangle \ll \sigma$.
2. $s = \text{sign}(\Pi)$.
3. $\sigma \leq^s \tau$.
4. $\text{quantsign}(\tau, \Pi R^f \Sigma_j \Gamma) \neq -s$ for $1 \leq j \leq g$.
5. Σ_1 begins with L .
6. $\text{sign}(\Sigma_1) = -\text{sign}(\Sigma_2)$.

then there exists some prefix $\Delta < \Gamma$ such that

$$\langle X, \Pi R^f \cdot \{\Sigma_1, \dots, \Sigma_g\} \cdot \Delta \rangle \ll \tau$$

where X is either some type variable β or some path Φ such that $\Phi \leq \Pi R^f$ and $\text{sign}(\Phi) = s$.

Proof: Consider the sequence of rewrite steps which proves $\sigma \leq^s \tau$:

$$\sigma = \tau_1 \sqsubset^s \dots \sqsubset^s \tau_n = \tau$$

Suppose the desired conclusion does not hold. Let τ_{k+1} be the first type in the sequence which the desired leaf-group pattern does not match. Thus, for $1 \leq i \leq k$ there exist X_i and Δ_i such that

$$\langle X_i, \Pi R^f \cdot \{\Sigma_1, \dots, \Sigma_g\} \cdot \Delta_i \rangle \ll \tau_i \tag{4}$$

where X_i is either a type variable β_i or a path Φ_i such that $\Phi_i \leq \Pi R^f$ and $\text{sign}(\Phi_i) = s$ but for $i = k + 1$ there do not exist X_{k+1} and Δ_{k+1} satisfying (4).

The rewrite step $\tau_k \sqsubset^s \tau_{k+1}$ can not have used the (h-distr s) or (h-distr $-s$) rules. If this rewrite step used the rule (h-inst $-s$), then for some $j \in \{1, \dots, g\}$ it must be the case that $\text{quantsign}(\tau_{k+1}, \Pi R^f \Sigma_j \Gamma) = -s$, which would imply $\text{quantsign}(\tau_n, \Pi R^f \Sigma_j \Gamma) = -s$, contradicting requirement 4. Thus, the rule (h-inst s) must have been used like

$$\tau_k = \mu[\forall \alpha. \pi] \sqsubset^s \mu[\pi[\alpha := \rho]] = \tau_{k+1}$$

where the quantifier “ $\forall \alpha$ ” corresponds to the leaf-group pattern in (4) and $\text{hole}(\mu[\]) = \Phi_i$. (X_k can not be a type variable.)

Suppose $\text{quantsign}(\rho, \Delta_k \setminus \Gamma) = q \neq 0$. Then one of the values $\text{quantsign}(\tau_{k+1}, \Pi R^f \Sigma_1 \Gamma)$ or $\text{quantsign}(\tau_{k+1}, \Pi R^f \Sigma_2 \Gamma)$ must be $-s$, which implies that either $\text{quantsign}(\tau_n, \Pi R^f \Sigma_1 \Gamma)$ or $\text{quantsign}(\tau_n, \Pi R^f \Sigma_2 \Gamma)$ is $-s$, which contradicts requirement 4. Thus, it must hold that $\text{quantsign}(\rho, \Delta_k \setminus \Gamma) = 0$.

Define $\Delta_{k+1} = \Delta_k \cdot \text{leaf}(\rho, \Delta_k \setminus \Gamma)$. Suppose the type context $\mu[\]$ contains a quantifier which captures $\text{leafvar}(\rho, \Delta_k \setminus \Gamma)$. This quantifier must be at a position whose sign is s , since otherwise a contradiction with requirement 4 results. Let Φ_{k+1} be the position of this quantifier. Otherwise, the type context $\mu[\]$ does not capture $\text{leafvar}(\rho, \Delta_k \setminus \Gamma)$. Let $\beta_i = \text{leafvar}(\rho, \Delta_k \setminus \Gamma)$. In either of these cases, for $i = k + 1$ this satisfies (4), contradicting our assumption. Therefore, the desired result must hold for $\tau_n = \tau$. ■

The following lemma is a variation of Lemma 3.2. When we are concerned with some particular infinite paths, we can weaken the conditions under which two leaves belonging to the same leaf group can not be split by subtyping. If the first path is R^ω (respectively, $R^k L R^\omega$ if using the lemma in the reverse direction) and the second path ends with R^ω , then it is sufficient to require that the *quantsign* values for the second path be positive (respectively, negative).

Lemma 3.3 *If there exist types σ and τ , paths Π and Δ , and numbers f and g such that the following requirements are met:*

1. $\langle \Pi, \Pi R^f \cdot \{R^g, L\Delta\} \rangle \ll \sigma$.
2. $s = \text{sign}(\Pi)$.
3. $\sigma \leq^s \tau$
4. $\text{quantsign}(\tau, \Pi R^f L \Delta R^\omega) = s$.
5. $\text{sign}(\Delta) = +1$.
6. Π contains either zero or one L .

then $\langle \Phi, \Pi R^f \cdot \{R^g R^j, L \Delta R^j\} \rangle \ll \tau$ for some Φ , h , and j where $\Phi R^h = \Pi R^f$.

Proof: Consider the sequence of rewrite steps which proves $\sigma \leq^s \tau$:

$$\sigma = \tau_1 \sqsubset^s \dots \sqsubset^s \tau_n = \tau$$

Suppose the desired conclusion does not hold. Let τ_{k+1} be the first type in the sequence which the desired leaf-group pattern does not match. Thus, for $1 \leq i \leq k$ there exist Φ_i , h_i , and j_i such that

$$\langle \Phi_i, \Phi_i R^{h_i} \cdot \{R^g R^{j_i}, L \Delta R^{j_i}\} \rangle \ll \tau_i \quad (5)$$

but for $i = k + 1$ there do not exist Φ_{k+1} , h_{k+1} , and j_{k+1} satisfying (5).

The rewrite step $\tau_k \sqsubset^s \tau_{k+1}$ can not have used the (h-distr s) or (h-distr $-s$) rules. The only ways the rule (h-inst $-s$) could affect the leaf or quantifier positions on the paths ΠR^ω or $\Pi R^f L \Delta R^\omega$ would result in $\text{quantsign}(\tau_{k+1}, \Pi R^f L \Delta R^\omega) = -s$, which would imply $\text{quantsign}(\tau_n, \Pi R^f L \Delta R^\omega) = -s$, contradicting requirement 4. Thus, the rule (h-inst s) must have been used like

$$\tau_k = \mu[\forall \alpha. \pi] \sqsubset^s \mu[\pi[\alpha := \rho]] = \tau_{k+1}$$

where the quantifier “ $\forall \alpha$ ” corresponds to the leaf-group pattern in (5) and $\text{hole}(\mu[\]) = \Phi_i$.

If $\text{quantsign}(\rho, R^\omega) = +1$, then $\text{quantsign}(\tau_{k+1}, \Pi R^f L \Delta R^\omega) = -s$, which implies that $\text{quantsign}(\tau_n, \Pi R^f L \Delta R^\omega) = -s$, contradicting requirement 4. Thus, $\text{quantsign}(\rho, R^\omega) = 0$. It must be the case that the type context $\mu[\]$ contains a quantifier at a position whose sign is s which captures $\text{leafvar}(\rho, R^\omega)$. Let Φ_{k+1} be the position of this quantifier, define h_{k+1} so that $\Phi_{k+1} R^{h_{k+1}} = \Pi R^f$, and define $j_{k+1} = j_k + |\text{leaf}(\rho, R^\omega)|$. Then for $i = k + 1$ this satisfies (5), contradicting our assumption. Therefore, the desired result must hold for $\tau_n = \tau$. ■

Another variation of Lemma 3.2 is the following lemma. When we know that the size of the type does not change, at least for the particular paths we are concerned with, then we can weaken the conditions under which a leaf group must be preserved. If the paths can not get longer and at least one leaf is negative (when the quantifier is at a positive position), then it is not necessary to require that another leaf be positive.

Lemma 3.4 *If there exist types σ and τ , paths $\Pi, \Sigma_1, \dots, \Sigma_g \in \mathcal{P}$, and numbers e and f such that the following requirements are met:*

1. $\langle \Pi, \text{PR}^f \cdot \{\Sigma_1, \dots, \Sigma_g\} \rangle \triangleleft \sigma$.
2. $s = \text{sign}(\Pi)$.
3. $\sigma \leq^s \tau$.
4. $\text{quantsign}(\tau, \text{PR}^f \Sigma_j \Gamma) \neq -s$ for $1 \leq j \leq g$ and $\Gamma \in \mathcal{P}^\omega$.
5. Σ_1 begins with L .
6. $\text{sign}(\Sigma_e) = -s$.
7. $\text{leaf}(\tau, \text{PR}^f \Sigma_e) = \text{PR}^f \Sigma_e$.

then

$$\langle X, \text{PR}^f \cdot \{\Sigma_1, \dots, \Sigma_g\} \rangle \triangleleft \tau$$

where X is either some type variable β or some path Φ such that $\Phi \leq \text{PR}^f$ and $\text{sign}(\Phi) = s$.

Proof: Consider the sequence of rewrite steps which proves $\sigma \leq^s \tau$:

$$\sigma = \tau_1 \sqsubset^s \dots \sqsubset^s \tau_n = \tau$$

Suppose the desired conclusion does not hold. Let τ_{k+1} be the first type in the sequence which the desired leaf-group pattern does not match. Thus, for $1 \leq i \leq k$ there exists X_i such that

$$\langle X_i, \text{PR}^f \cdot \{\Sigma_1, \dots, \Sigma_g\} \rangle \triangleleft \tau_i \tag{6}$$

where X_i is either a type variable β_i or a path Φ_i such that $\Phi_i \leq \text{PR}^f$ and $\text{sign}(\Phi_i) = s$ but for $i = k + 1$ there do not exist X_{k+1} and Δ_{k+1} satisfying (6).

The rewrite step $\tau_k \sqsubset^s \tau_{k+1}$ can not have used the (h-distr s) or (h-distr $-s$) rules, because there is no way they could make τ_{k+1} not satisfy (6) without violating requirement 4. If this rewrite step used the rule (h-inst $-s$), then for some $j \in \{1, \dots, g\}$ and some $\Gamma \in \mathcal{P}^\omega$ it must be the case that $\text{quantsign}(\tau_{k+1}, \text{PR}^f \Sigma_j \Gamma) = -s$, which would imply $\text{quantsign}(\tau_n, \text{PR}^f \Sigma_j \Gamma) = -s$, contradicting requirement 4. Thus, the rule (h-inst s) must have been used like

$$\tau_k = \mu[\forall \alpha. \pi] \sqsubset^s \mu[\pi[\alpha := \rho]] = \tau_{k+1}$$

where the quantifier “ $\forall \alpha$ ” corresponds to the leaf-group pattern in (6) and $\text{hole}(\mu[\])$ = Φ_i . (X_k can not be a type variable.)

Suppose $\text{quantsign}(\rho, \Gamma) = q \neq 0$. Then one of the values $\text{quantsign}(\tau_{k+1}, \text{PR}^f \Sigma_1 \Gamma)$ or $\text{quantsign}(\tau_{k+1}, \text{PR}^f \Sigma_2 \Gamma)$ must be $-s$, which implies that either $\text{quantsign}(\tau_n, \text{PR}^f \Sigma_1 \Gamma)$ or $\text{quantsign}(\tau_n, \text{PR}^f \Sigma_2 \Gamma)$ is $-s$, which contradicts requirement 4. Thus, it must hold that $\text{quantsign}(\rho, \Delta_k \setminus \Gamma) = 0$. Suppose $\text{leaf}(\rho, \Gamma) = \Gamma' \neq \varepsilon$. This implies that

$$\text{leaf}(\tau_{k+1}, \text{PR}^f \Sigma_e \Gamma) > \text{leaf}(\tau_n, \text{PR}^f \Sigma_e \Gamma)$$

which implies that $\text{quantsign}(\tau_n, \text{PR}^f \Sigma_e \Gamma) = -s$, contradicting requirement 4. Thus, by elimination, $\rho = \alpha$ for some type variable α .

Suppose the type context $\mu[\]$ contains a quantifier which captures α . This quantifier must be at a position whose sign is s , since otherwise a contradiction with requirement 4 results. Let Φ_{k+1} be the position of this quantifier. Otherwise, let $\beta_i = \alpha$. In either case, for $i = k + 1$ this satisfies (6), contradicting our assumption. Therefore, the desired result must hold for $\tau_n = \tau$. ■

The next lemma states that when the conditions of Lemma 3.2 are not met, we can still know something about how subtyping can change the relative lengths of certain paths.

Lemma 3.5 *If $\sigma \leq^s \tau$ and $\langle \Pi, \{\Sigma_1, \Sigma_2\} \rangle \triangleleft \sigma$ and $\text{sign}(\Pi) = s$ and $\text{sign}(\Sigma_1) \neq \text{sign}(\Sigma_2)$ and $\text{quantsign}(\tau, \Sigma_1\Gamma) \neq -s$, then*

$$|\text{leaf}(\tau, \Sigma_1\Gamma)| - |\Sigma_1| \geq |\text{leaf}(\tau, \Sigma_2\Gamma)| - |\Sigma_2|$$

Proof: Consider the sequence of rewrite steps which proves $\sigma \leq^s \tau$:

$$\sigma = \tau_1 \sqsubset^s \cdots \sqsubset^s \tau_n = \tau$$

For $1 \leq i \leq n$, it must be the case that $\text{quantsign}(\tau_i, \Sigma_1\Gamma) \neq -s$. If $\text{quantsign}(\tau_i, \Sigma_2\Gamma) \neq -s$ for $1 \leq i \leq n$, then the desired conclusion holds by Lemma 3.2. Otherwise, let τ_{k+1} be the first type in the sequence for which $\text{quantsign}(\tau_i, \Sigma_1\Gamma) \neq -s$. By Lemma 3.2, it holds that $\langle X_k, \{\Sigma_1, \Sigma_2\} \cdot \Delta_k \rangle \triangleleft \tau_k$ where $\Delta_k < \Gamma$ and for some X . By cases on the rule used to prove $\tau_k \sqsubset^s \tau_{k+1}$:

1. (h-inst s) $\text{leaf}(\tau_{k+1}, \Sigma_i\Gamma) = \Sigma_i\Delta_{k+1}$ for $i \in \{1, 2\}$ and for some $\Delta_{k+1} < \Gamma$.
2. (h-inst $-s$) $\text{leaf}(\tau_{k+1}, \Sigma_1\Gamma) = \Sigma_1\Delta_{k+1}$ and $\text{leaf}(\tau_{k+1}, \Sigma_2\Gamma) \leq \Sigma_2\Delta_{k+1}$ where $\Delta_{k+1} = \Delta_k$.
3. (h-distr s) $\text{leaf}(\tau_{k+1}, \Sigma_i\Gamma) = \Sigma_i\Delta_{k+1}$ for $i \in \{1, 2\}$ where $\Delta_{k+1} = \Delta_k$.
4. (h-distr $-s$) Not applicable.

Since $\text{quantsign}(\tau, \Sigma_1\Gamma) \neq -s$, it holds that $\text{leaf}(\tau, \Sigma_1\Gamma) \geq \Sigma_1\Delta_{k+1}$. Since it must be the case that $\text{quantsign}(\tau, \Sigma_2\Gamma) = -s$, it holds that $\text{leaf}(\tau, \Sigma_2\Gamma) \leq \Sigma_2\Delta_{k+1}$. Thus,

$$|\text{leaf}(\tau, \Sigma_1\Gamma)| - |\Sigma_1| \geq |\Delta_{k+1}| \geq |\text{leaf}(\tau, \Sigma_2\Gamma)| - |\Sigma_2|$$

■

The following lemma is vital for showing that a particular path in a type can not be longer than a certain length, or else a contradiction can be shown.

Lemma 3.6 *If the following conditions are true:*

1. $\sigma \leq \tau$.
2. $\text{quant}(\sigma, \Pi R^g L \Sigma R^\omega) = \Pi R^f$.
3. $\text{leaf}(\sigma, \Pi R^g L \Sigma R^\omega) \geq \Pi R^g L \Sigma$.
4. $\text{quant}(\tau, \Pi R^g L \Sigma R^\omega) \not\leq \Pi R^g$.
5. $\text{sign}(\Pi) = +1 = \text{sign}(\Sigma)$.

then it must be the case that $\text{quantsign}(\tau, \Pi R^g L \Sigma R^\omega) \neq +1$.

Proof: Consider the sequence of rewrite steps which proves $\sigma \leq \tau$:

$$\sigma = \tau_1 \sqsubset \cdots \sqsubset \tau_n = \tau$$

If for $1 \leq i \leq n$ it holds that

$$\mathit{quantsign}(\tau_i, \Pi R^g L \Sigma R^\omega) = +1 \quad (7)$$

$$\mathit{quant}(\tau_i, \Pi R^g L \Sigma R^\omega) \leq \Pi R^g \quad (8)$$

$$\mathit{leaf}(\tau_i, \Pi R^g L \Sigma R^\omega) \geq \Pi R^g L \Sigma \quad (9)$$

then the claim of the lemma is true because (8) contradicts one of the hypotheses of the claim. Otherwise, let τ_{k+1} be the first type in the sequence such that (7), (8), and (9) are not satisfied for $i = k + 1$. If this is the case because (7) is false, then the claim of the lemma is true, because this implies $\mathit{quantsign}(\tau, \Pi R^g L \Sigma R^\omega) \neq +1$. If it is because (9) is false, then this implies (7) is false, which implies the claim of the lemma.

Otherwise, suppose that

$$\mathit{quantsign}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = +1 \quad (10)$$

$$\mathit{leaf}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) \geq \Pi R^g L \Sigma \quad (11)$$

By elimination, it must hold that

$$\mathit{quant}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) \not\leq \Pi R^g \quad (12)$$

Consider the rewrite step $\tau_k \sqsubset \tau_{k+1}$ by cases on the rule used to prove it.

1. (h-inst +1) The quantifier for the leaf on the path $\Pi R^g L \Sigma R^\omega$ must have been instantiated like this:

$$\tau_k = \pi[\forall\alpha.\varphi] \sqsubset \pi[\varphi[\alpha:=\rho]]$$

Suppose $\mathit{quantsign}(\rho, R^\omega) = +1$. Then $\mathit{quantsign}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = -1$, contradicting (10). Thus, $\mathit{quantsign}(\rho, R^\omega) = -1$. Let $\alpha = \mathit{leafvar}(\rho, R^\omega)$. If $\alpha \in \text{BHV}(\pi[\])$, then this contradicts (12). However, if $\alpha \notin \text{BHV}(\pi[\])$, then it must be the case that $\mathit{quantsign}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = 0$, contradicting (10). Thus, the rule (h-inst +1) can not have been used.

2. (h-inst -1) There is no way the rule (h-inst -1) can be used to affect the path $\Pi R^g L \Sigma R^\omega$ without causing $\mathit{quantsign}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = -1$, which would contradict (10).
3. (h-distr +1) In order for a use of (h-distr +1) to cause (12), it must be the case that $\mathit{quant}(\tau_k, \Pi R^g L \Sigma R^\omega) = \Pi R^g$, which implies that $\mathit{quant}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = \Pi R^g L$, which implies that $\mathit{quantsign}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = -1$, which contradicts (10).
4. (h-distr -1) There is no way the rule (h-distr -1) can be used to affect the path $\Pi R^g L \Sigma R^\omega$ without causing $\mathit{quantsign}(\tau_{k+1}, \Pi R^g L \Sigma R^\omega) = -1$, which would contradict (10).

Thus, it is impossible for both (10) and (11) to be true. Therefore, in all cases the claim of the lemma is true. ■

All of the previous lemmas in this section refer only to the subtyping relation. The next step is to analyze the interaction of the subtyping rules with the λ -term typing rules. The next lemma applies the previous results to typings of actual λ -terms. The way λ -terms are constructed can be used to prove a number of useful properties about the kind of typings that can occur. The properties proven by this lemma are used pervasively throughout Section 5, often without any reference to the lemma.

Lemma 3.7 *Let \mathcal{D} be a derivation in $F+\eta$ that types a term P . Then the following properties are true:*

1. *If $M \subset P$ then*

- (a) *If $\text{quantsign}(\text{FDT}(M), \Pi) = +1$, then $\text{quantsign}(\text{IDT}(M), \Pi) = +1$ and $\text{leaf}(\text{FDT}(M), \Pi) \geq \text{leaf}(\text{IDT}(M), \Pi)$.*
- (b) *If $\text{quantsign}(\text{IDT}(M), \Pi) = -1$, then $\text{quantsign}(\text{FDT}(M), \Pi) = -1$ and $\text{leaf}(\text{IDT}(M), \Pi) \geq \text{leaf}(\text{FDT}(M), \Pi)$.*

2. *If $(\lambda w.M) \subset P$ then*

- (a) $L \setminus \text{leaf}(\text{IDT}(\lambda w.M), L\Pi) = \text{leaf}(\mathcal{G}(w), \Pi)$.
- (b) *If $\text{quantsign}(\mathcal{G}(w), \Pi) = +1$, then $\text{quantsign}((\lambda w.M), L\Pi) = -1$ and $\text{leaf}(\mathcal{G}(w), \Pi) \geq L \setminus \text{leaf}(\text{FDT}(\lambda w.M), L\Pi)$.*
- (c) *If $\text{quantsign}(\text{FDT}(M), \Pi) = -1$, then $\text{quantsign}((\lambda w.M), R\Pi) = -1$ and $\text{leaf}(\text{FDT}(M), \Pi) \geq R \setminus \text{leaf}(\text{FDT}(\lambda w.M), R\Pi)$.*
- (d) *If $\text{quantsign}(\text{IDT}(\lambda w.M), L\Pi) \neq -1$, then $\text{quantsign}(\mathcal{G}(w), \Pi) \neq +1$. If $\text{quantsign}(\text{FDT}(\lambda w.M), L\Pi) \neq -1$, then $L \setminus \text{leaf}(\text{FDT}(\lambda w.M), L\Pi) \geq \text{leaf}(\mathcal{G}(w), \Pi)$.*
- (e) *If $\text{quantsign}(\text{IDT}(\lambda w.M), R\Pi) \neq -1$, then $\text{quantsign}(\text{FDT}(M), \Pi) \neq -1$. If $\text{quantsign}(\text{FDT}(\lambda w.M), R\Pi) \neq -1$, then $R \setminus \text{leaf}(\text{FDT}(\lambda w.M), R\Pi) \geq \text{leaf}(\text{IDT}(M), \Pi)$.*
- (f) *If $\text{quantsign}(\text{IDT}(\lambda w.M), L\Pi) = +1$ and $\text{quantsign}(\mathcal{G}(w), \Pi) = 0$, then $\text{quant}(\text{IDT}(\lambda w.M), L\Pi) = \varepsilon$.*
- (g) *If $\text{quantsign}(\text{IDT}(\lambda w.M), R\Pi) = +1$ and $\text{quantsign}(\text{FDT}(M), \Pi) = 0$, then $\text{quant}(\text{IDT}(\lambda w.M), R\Pi) = \varepsilon$.*

3. *If $(MN) \subset P$ then*

- (a) $\text{quantsign}(\text{FDT}(M), L\Pi) = -\text{quantsign}(\text{FDT}(N), \Pi) \neq -1$.
- (b) $\text{quantsign}(\text{FDT}(M), L\Pi) \neq +1$ implies $L \setminus \text{leaf}(\text{FDT}(M), L\Pi) \geq \text{leaf}(\text{IDT}(N), \Pi)$.
- (c) $\text{quantsign}(M, L\Pi) \neq +1$ implies $L \setminus \text{leaf}(\text{IDT}(M), L\Pi) \geq \text{leaf}(\text{IDT}(N), \Pi)$.
- (d) $\text{quantsign}(\text{FDT}(N), \Pi) \neq +1$ implies $L \setminus \text{leaf}(\text{IDT}(M), L\Pi) \leq \text{leaf}(\text{FDT}(N), \Pi)$.
- (e) $\text{quantsign}(N, \Pi) \neq +1$ implies $L \setminus \text{leaf}(\text{IDT}(M), L\Pi) \leq \text{leaf}(\text{IDT}(N), \Pi)$.
- (f) *If $\text{quantsign}(\text{FDT}(M), R\Pi) = -1$, then $\text{quantsign}((MN), \Pi) = -1$ and $R \setminus \text{leaf}(\text{FDT}(M), R\Pi) \geq \text{leaf}(\text{FDT}(MN), \Pi)$.*
- (g) *If $\text{quantsign}(\text{IDT}(MN), \Pi) = +1$, then $\text{quantsign}(M, R\Pi) = +1$ and $R \setminus \text{leaf}(\text{FDT}(M), R\Pi) \leq \text{leaf}(\text{FDT}(MN), \Pi)$.*

4. *If $((\lambda w.\square)N) \subset P$ then*

- (a) $\text{quantsign}(\mathcal{G}(w), \Pi) = +1$ implies that $\text{quantsign}(N, \Pi) = +1$ and $\text{leaf}(\mathcal{G}(w), \Pi) \geq \text{leaf}(\text{FDT}(N), \Pi) \geq \text{leaf}(\text{IDT}(N), \Pi)$.
- (b) $\text{quantsign}(\text{FDT}(N), \Pi) = -1$ implies that $\text{quantsign}(\mathcal{G}(w), \Pi) \neq +1$ and $\text{leaf}(\text{FDT}(N), \Pi) \geq \text{leaf}(\mathcal{G}(w), \Pi)$.
- (c) $\text{quantsign}(N, \Pi) = -1$ implies that $\text{quantsign}(\mathcal{G}(w), \Pi) \neq +1$ and $\text{leaf}(\text{IDT}(N), \Pi) \geq \text{leaf}(\mathcal{G}(w), \Pi)$.

5. If $((\lambda w.\square)(\lambda v.\square)) \subset P$, then at most one of the values $\text{quantsign}(\mathcal{G}(w), L\Pi)$ and $\text{quantsign}(\mathcal{G}(v), \Pi)$ can be positive. Furthermore:
- (a) If $\text{quantsign}(\mathcal{G}(w), L\Pi) = +1$ then $L \setminus \text{leaf}(\mathcal{G}(w), L\Pi) \geq \text{leaf}(\mathcal{G}(v), \Pi)$.
 - (b) If $\text{quantsign}(\mathcal{G}(v), \Pi) = +1$ then $L \setminus \text{leaf}(\mathcal{G}(w), L\Pi) \leq \text{leaf}(\mathcal{G}(v), \Pi)$.
6. If $((w\square_1 \dots \square_n)(w\square_1 \dots \square_n)) \subset P$, then $\text{quantsign}(\mathcal{G}(w), R^n L^\omega) = +1$.

Proof: By Lemma 3.1, using the results of each property to prove the subsequent properties. ■

4 Positive and Negative Subtyping

This section focuses on the rewriting-style rules for subtyping. By careful analysis, it is shown that particular rewrite sequences can be reordered so that the rules are used in a more convenient order. Subtyping can be split into rewrites that occur at positive and negative locations, and all of the positive rewrites can be performed first.

Definition 4.1 (Active Leaf Group of Rewrite Step) For a rewrite step $\pi \sqsubset^s \mu$, the rule used determines the rewrite step's *active leaf group*. If (h-inst s) is used like

$$\pi = \tau[\forall\alpha.\sigma] \sqsubset^s \tau[\sigma[\alpha:=\rho]] = \mu \quad \text{where } s = \text{holesign}(\tau[\])$$

then the active leaf group of the rewrite step is $\langle \Pi, \Pi \cdot P \rangle \in \pi$ where $\Pi = \text{hole}(\tau[\])$ and $\langle \alpha, P \rangle \in \sigma$. If (h-distr s) is used like

$$\pi = \tau[\forall\alpha.(\sigma \rightarrow \rho)] \sqsubset^s \tau[(\forall\alpha.\sigma) \rightarrow (\forall\alpha.\rho)] = \mu \quad \text{where } s = \text{holesign}(\tau[\])$$

then the active leaf group of the rewrite step is $\langle \Pi, \Pi \cdot P \rangle \in \pi$ where $\Pi = \text{hole}(\tau[\])$ and $\langle \alpha, P \rangle \in \sigma \rightarrow \rho$.

Definition 4.2 (Residual of Leaf Group after Rewrite Step) After a rewrite step $\pi \sqsubset \mu$, the *residual* in μ of a leaf group $\langle X, P \rangle \in \pi$ is defined as follows. If there is a Y such that $\langle Y, P \rangle \in \mu$, then this is the residual. Otherwise, the definition of the residual depends on the rule used to prove the rewrite step $\pi \sqsubset \mu$.

1. (h-inst +1), (h-distr +1). If $\langle X, P \rangle$ is the active leaf group of the rewrite step, then it has no residual. Otherwise, there must be a $Q \supseteq P$ such that $\langle X, Q \rangle \in \pi$, which is then the residual.
2. (h-inst -1), (h-distr -1). If there is a $Q \subseteq P$ such that $\langle X, Q \rangle \in \pi$ and this is not the active leaf group of the rewrite step, then this is the residual. Otherwise, there is no residual.

Definition 4.3 (Positive and Negative Subtyping) If $\pi \sqsubset \mu$ is proven using either the rule (h-inst +1) or the rule (h-distr +1), then we say that $\pi \sqsubset \mu$ by *positive subtyping*, written $\pi \boxplus \mu$. Similarly, using only rules (h-inst -1) and (h-distr -1) is *negative subtyping*, written $\pi \boxminus \mu$.

Lemma 4.4 If $\sigma_1 \boxplus \sigma_2 \boxplus \sigma_3$, then there exists a type σ_4 such that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$. In other words, *negative subtyping can be postponed after positive subtyping*.

Proof: Let $\sigma_1 \boxplus \sigma_2$ be proven by rule $R_1 \in \{(\text{h-inst } -1), (\text{h-distr } -1)\}$ with active leaf group $\langle \Pi_1, P_1 \rangle \in \sigma_2$ and $\sigma_2 \boxplus \sigma_3$ by rule $R_2 \in \{(\text{h-inst } +1), (\text{h-distr } +1)\}$ with active leaf group $\langle \Pi_2, P_2 \rangle \in \sigma_2$. By cases on the rules R_1 and R_2 .

1. R_1 is (h-inst -1) and R_2 is (h-inst $+1$). By cases on whether $\Pi_1 < \Pi_2$, $\Pi_1 > \Pi_2$, or $\Pi_1 \not\prec \Pi_2 \not\prec \Pi_1$.

(a) ($\Pi_1 < \Pi_2$) Let

$$\sigma_1 = \pi_1[S_1(\pi_2[\forall\alpha_2.\tau])], \quad \sigma_2 = \pi_1[\forall\alpha_1.\pi_2[\forall\alpha_2.\tau]], \quad \sigma_3 = \pi_1[\forall\alpha_1.\pi_2[S_2(\tau)]]$$

for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, type variables α_1 and α_2 , type τ , and substitutions S_1 , and S_2 . Let $\sigma_4 = \pi_1[S_1(\pi_2[S_2(\tau)])]$. It is immediate that $\sigma_4 \in \sigma_3$, but it requires some work to check that $\sigma_1 \in \sigma_4$. We may assume without loss of generality that $\alpha_1 \notin \text{FTV}(S_1(\alpha_1))$ and that $\text{BHV}(\pi_2[\forall\alpha_2.[\]]) \cap \text{FTV}(S_1(\alpha_1)) = \emptyset$. Thus, $S_1(\pi_2[\forall\alpha_2.\tau]) = S_1(\pi_2[\forall\alpha_2.S_1(\tau)])$. Let S_3 be the substitution $[\alpha_2 := S_1(S_2(\alpha_2))]$, implying $S_3(S_1(\tau)) = S_1(S_2(\tau))$. Thus, we know this:

$$\sigma_1 = \pi_1[S_1(\pi_2[\forall\alpha_2.S_1(\tau)])] \in \pi_1[S_1(\pi_2[S_3(S_1(\tau))])] = \pi_1[S_1(\pi_2[S_1(S_2(\tau))])] = \sigma_4$$

(b) ($\Pi_1 > \Pi_2$) By similar reasoning to the case where ($\Pi_1 < \Pi_2$), except some aspects are reversed.

(c) ($\Pi_1 \not\prec \Pi_2 \not\prec \Pi_1$) Let $\sigma_2 = \pi[\forall\alpha_1.\tau_1, \forall\alpha_2.\tau_2]$, $\sigma_1 = \pi[S_1(\tau_1), \forall\alpha_2.\tau_2]$, and $\sigma_3 = \pi[\forall\alpha_1.\tau_1, S_2(\tau_2)]$ for appropriate type context $\pi[\]$, type variables α_1 and α_2 , types τ_1 and τ_2 , and substitutions S_1 , and S_2 . Let $\sigma_4 = \pi[S_1(\tau_1), S_2(\tau_2)]$. Then $\sigma_1 \in \sigma_4 \in \sigma_3$.

2. R_1 is (h-inst -1) and R_2 is (h-distr $+1$). By cases on whether $\Pi_1 < \Pi_2$, $\Pi_1 > \Pi_2$, or $\Pi_1 \not\prec \Pi_2 \not\prec \Pi_1$.

(a) ($\Pi_1 < \Pi_2$) Let $\sigma_2 = \pi_1[\forall\alpha_1.\pi_2[\forall\alpha_2.(\tau_1 \rightarrow \tau_2)]]$, $\sigma_1 = \pi_1[S_1(\pi_2[\forall\alpha_2.(\tau_1 \rightarrow \tau_2)])]$, and $\sigma_3 = \pi_1[\forall\alpha_1.\pi_2[(\forall\alpha_2.\tau_1) \rightarrow (\forall\alpha_2.\tau_2)]]$ for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 and τ_2 , type variables α_1 and α_2 , and substitution S_1 . Let $\sigma_4 = \pi_1[S_1(\pi_2[(\forall\alpha_2.\tau_1) \rightarrow (\forall\alpha_2.\tau_2)])]$. It is immediate that $\sigma_4 \in \sigma_3$. Assume $\alpha_1 \notin \text{FTV}(S_1(\alpha_1))$ and that $\text{BHV}(\pi_2[\forall\alpha_2.[\]]) \cap \text{FTV}(S_1(\alpha_1)) = \emptyset$. Thus,

$$\begin{aligned} \sigma_1 &= \pi_1[S_1(\pi_2[\forall\alpha_2.(S_1(\tau_1) \rightarrow S_1(\tau_2))])] \\ &\in \pi_1[S_1(\pi_2[(\forall\alpha_2.S_1(\tau_1)) \rightarrow (\forall\alpha_2.S_1(\tau_2))])] = \sigma_4 \end{aligned}$$

(b) ($\Pi_1 \geq \Pi_2 L$) Let $\sigma_2 = \pi_2[\forall\alpha_2.(\pi_1[\forall\alpha_1.\tau_1] \rightarrow \tau_2)]$, $\sigma_1 = \pi_2[\forall\alpha_2.(\pi_1[S_1(\tau_1)] \rightarrow \tau_2)]$, and $\sigma_3 = \pi_2[(\forall\alpha_2.\pi_1[\forall\alpha_1.\tau_1]) \rightarrow (\forall\alpha_2.\tau_2)]$, for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 and τ_2 , type variables α_1 and α_2 , and substitution S_1 . Let $\sigma_4 = \pi_2[(\forall\alpha_2.\pi_1[S_1(\tau_1)]) \rightarrow (\forall\alpha_2.\tau_2)]$. It holds that $\sigma_1 \in \sigma_4 \in \sigma_3$.

(c) ($\Pi_1 \geq \Pi_2 R$) By similar reasoning to case where $\Pi_1 \geq \Pi_2 L$.

(d) ($\Pi_1 \not\prec \Pi_2 \not\prec \Pi_1$) Easy.

3. R_1 is (h-distr -1) and R_2 is (h-inst $+1$). By reasoning identical to the case where R_1 is (h-inst -1) and R_2 is (h-distr $+1$).

4. R_1 is (h-distr -1) and R_2 is (h-distr $+1$). Let $P_i^\Delta = \{\Sigma \in P_i \mid \Pi_i \Delta \leq \Sigma\}$ for $i \in \{1, 2\}$ and $\Delta \in \{L, R\}$. It is the case that

$$\begin{aligned} \sigma_1 &= (\sigma_2 - \{\langle \Pi_1, P_1 \rangle\}) \cup \{\langle \Pi_1 L, P_1^L \rangle, \langle \Pi_1 R, P_1^R \rangle\} \\ \sigma_3 &= (\sigma_2 - \{\langle \Pi_2, P_2 \rangle\}) \cup \{\langle \Pi_2 L, P_2^L \rangle, \langle \Pi_2 R, P_2^R \rangle\} \end{aligned}$$

Let $\sigma_4 = (\sigma_2 - \{\langle \Pi_1, P_1 \rangle, \langle \Pi_2, P_2 \rangle\}) \cup \{\langle \Pi_1 L, P_1^L \rangle, \langle \Pi_1 R, P_1^R \rangle, \langle \Pi_2 L, P_2^L \rangle, \langle \Pi_2 R, P_2^R \rangle\}$. Then $\sigma_1 \in \sigma_4 \in \sigma_3$.

■

Lemma 4.5 For all paths $\Sigma_1, \Sigma_2 \in \mathcal{P}$, sets of paths $P_1, P_2 \subset \mathcal{P}$, types $\sigma_1, \sigma_2, \sigma_3 \in \mathbb{T}$, and rules $R_1, R_2 \in \{(\text{h-inst } +1), (\text{h-distr } +1)\}$, if it is the case that

1. $\langle \Sigma_1, P_1 \rangle$ and $\langle \Sigma_2, P_2 \rangle$ are distinct leaf groups in σ_1 .
2. $\sigma_1 \boxplus \sigma_2$ by rule R_1 where $\langle \Sigma_1, P_1 \rangle$ is the active leaf group.
3. $\langle \Sigma_2, P_2' \rangle \in \sigma_2$ is the residual of $\langle \Sigma_2, P_2 \rangle \in \sigma_1$.
4. $\sigma_2 \boxplus \sigma_3$ by rule R_2 where $\langle \Sigma_2, P_2' \rangle$ is the active leaf group.
5. If R_1 is $(\text{h-inst } +1)$ and R_2 is $(\text{h-distr } +1)$, then $\Sigma_1 \neq \Sigma_2$.

then there must exist a type σ_4 such that

6. $\sigma_1 \boxplus \sigma_4$ by rule R_2 where $\langle \Sigma_2, P_2 \rangle$ is the active leaf group.
7. $\langle \Sigma_1, P_1' \rangle \in \sigma_4$ is the residual of $\langle \Sigma_1, P_1 \rangle \in \sigma_1$.
8. $\sigma_4 \boxplus \sigma_3$ by rule R_1 where $\langle \Sigma_1, P_1' \rangle$ is the active leaf group.

In other words, the use of rule R_1 on $\langle \Sigma_1, P_1 \rangle$ can almost always (except when condition 5 is true) be postponed after the use of rule R_2 on (the residual of) $\langle \Sigma_2, P_2 \rangle$.

Proof: By cases on the rules R_1 and R_2 .

1. R_1 is $(\text{h-inst } +1)$ and R_2 is $(\text{h-inst } +1)$. By cases on whether $\Sigma_1 = \Sigma_2$, $\Sigma_1 < \Sigma_2$, $\Sigma_1 > \Sigma_2$, or $\Sigma_1 \not\leq \Sigma_2 \not\leq \Sigma_1$.

(a) ($\Sigma_1 = \Sigma_2$) Let $\sigma_1 = \pi[\forall\alpha_2.\forall\alpha_1.\tau]$, $\sigma_2 = \pi[\forall\alpha_2.S_1(\tau)]$, and $\sigma_3 = \pi[S_2(S_1(\tau))]$ for appropriate type context $\pi[\]$, type τ , type variables α_1 and α_2 , and substitutions S_1 and S_2 . Initial α -conversion lets us assume that $\alpha_1 \notin \text{FTV}(S_1(\alpha_1))$ and $\alpha_1, \alpha_2 \notin \text{FTV}(S_2(\alpha_2))$. Let $\sigma_4 = \pi[S_2(\forall\alpha_1.\tau)] = \pi[\forall\alpha_1.S_2(\tau)]$. It is immediate that $\sigma_1 \boxplus \sigma_4$. Let S_3 be $[\alpha_1 := S_2(S_1(\alpha_1))]$. Thus, $S_3(S_2(\tau)) = S_2(S_3(\tau)) = S_2(S_1(\tau))$. It holds that $\sigma_4 \boxplus \pi[S_3(S_2(\tau))] = \sigma_3$.

(b) ($\Sigma_1 < \Sigma_2$) Let $\sigma_1 = \pi_1[\forall\alpha_1.\pi_2[\forall\alpha_2.\tau]]$, and $\sigma_2 = \pi_1[S_1(\pi_2[\forall\alpha_2.\tau])]$ for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, type τ , type variables α_1 and α_2 , and substitution S_1 . It is safe to assume that $\text{BHV}(\pi_2[\forall\alpha_2.[\]]) \cap \text{FTV}(S_1(\alpha_1)) = \emptyset$ and that $\alpha_1 \notin \text{FTV}(S_1(\alpha_1))$. Thus, $\sigma_2 = \pi_1[S_1(\pi_2[\forall\alpha_2.S_1(\tau)])]$ Let $\sigma_3 = \pi_1[S_1(\pi_2[S_2(S_1(\tau))])]$ for appropriate substitution S_2 . We may assume that $\alpha_1 \notin \text{FTV}(S_2(\alpha_2))$. Thus, $S_1(S_2(\tau)) = S_2(S_1(\tau))$. Let $\sigma_4 = \pi_1[\forall\alpha_1.\pi_2[S_2(\tau)]]$. It is clear that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.

(c) ($\Sigma_1 > \Sigma_2$) Let

$$\sigma_1 = \pi_2[\forall\alpha_2.\pi_1[\forall\alpha_1.\tau]], \quad \sigma_2 = \pi_2[\forall\alpha_2.\pi_1[S_1(\tau)]], \quad \sigma_3 = \pi_2[S_2(\pi_1[S_1(\tau)])]$$

for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, type τ , type variables α_1 and α_2 , and substitutions S_1 and S_2 . Let $\sigma_4 = \pi_2[S_2(\pi_1[\forall\alpha_1.\tau])]$. It is immediate that $\sigma_1 \boxplus \sigma_4$. It is safe to assume that $\text{BHV}(\pi_1[\]) \cap \text{FTV}(S_2(\alpha_2)) = \emptyset$ and that $\alpha_1, \alpha_2 \notin \text{FTV}(S_2(\alpha_2))$. Thus, $\sigma_4 = \pi_2[S_2(\pi_1[\forall\alpha_1.S_2(\tau)])]$ and $\sigma_3 = \pi_2[S_2(\pi_1[S_2(S_1(\tau))])]$. Let S_3 be $[\alpha_1 := S_2(S_1(\alpha_1))]$. Thus, $S_3(S_2(\tau)) = S_2(S_1(\tau))$. Thus, $\sigma_4 \boxplus \pi_2[S_2(\pi_1[S_3(S_2(\tau))])] = \sigma_3$.

(d) ($\Sigma_1 \not\leq \Sigma_2 \not\leq \Sigma_1$) Easy.

2. R_1 is (h-inst +1) and R_2 is (h-distr +1). Condition 5 rules out the case $\Sigma_1 = \Sigma_2$. By cases on whether $\Sigma_1 < \Sigma_2$, $\Sigma_1 > \Sigma_2$, or $\Sigma_1 \not\prec \Sigma_2 \not\prec \Sigma_1$.

(a) ($\Sigma_1 < \Sigma_2$) Let $\sigma_1 = \pi_1[\forall\alpha_1.\pi_2[\forall\alpha_2.(\tau_1 \rightarrow \tau_2)]]$ and $\sigma_2 = \pi_1[S_1(\pi_2[\forall\alpha_2.(\tau_1 \rightarrow \tau_2)])]$ for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 and τ_2 , type variables α_1 and α_2 , and substitution S_1 . Note that $\pi_2[\]$ must have at least one “ \rightarrow ” in it, implying that there is no possibility that “ $\forall\alpha_2$ ” can capture any variables in $\text{FTV}(S_1(\alpha_1))$. We can safely assume that $\text{BHV}(\pi_2[\forall\alpha_2[\]]) \cap \text{FTV}(S_1(\alpha_1)) = \emptyset$ and that $\alpha_1 \notin \text{FTV}(S_1(\alpha_1))$. Thus, $\sigma_2 = \pi_1[S_1(\pi_2[\forall\alpha_2.(S_1(\tau_1) \rightarrow S_1(\tau_2))])]$. Let

$$\begin{aligned}\sigma_3 &= \pi_1[S_1(\pi_2[(\forall\alpha_2.S_1(\tau_1)) \rightarrow (\forall\alpha_2.S_1(\tau_2))])] \\ &= \pi_1[S_1(\pi_2[(\forall\alpha_2.\tau_1) \rightarrow (\forall\alpha_2.\tau_2)])]\end{aligned}$$

Let $\sigma_4 = \pi_1[\forall\alpha_1.\pi_2[(\forall\alpha_2.\tau_1) \rightarrow (\forall\alpha_2.\tau_2)]]$. It is immediate that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.

(b) ($\Sigma_1 > \Sigma_2$) Assume that $\Sigma_1 \geq \Sigma_2 R$. (The case where $\Sigma_1 \geq \Sigma_2 L$ is similar.) Let

$$\begin{aligned}\sigma_1 &= \pi_2[\forall\alpha_2.(\tau_2 \rightarrow \pi_1[\forall\alpha_1.\tau_1])] \\ \sigma_2 &= \pi_2[\forall\alpha_2.(\tau_2 \rightarrow \pi_1[S_1(\tau_1)])] \\ \sigma_3 &= \pi_2[(\forall\alpha_2.\tau_2) \rightarrow (\forall\alpha_2.\pi_1[S_1(\tau_1)])]\end{aligned}$$

for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 and τ_2 , type variables α_1 and α_2 , and substitution S_1 . Let $\sigma_4 = \pi_2[(\forall\alpha_2.\tau_2) \rightarrow (\forall\alpha_2.\pi_1[\forall\alpha_1.\tau_1])]$. It is immediate that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.

(c) ($\Sigma_1 \not\prec \Sigma_2 \not\prec \Sigma_1$) Easy.

3. R_1 is (h-distr +1) and R_2 is (h-inst +1). By cases on whether $\Sigma_1 < \Sigma_2$, $\Sigma_1 \geq \Sigma_2$, or $\Sigma_1 \not\prec \Sigma_2 \not\prec \Sigma_1$.

(a) ($\Sigma_1 < \Sigma_2$) Assume that $\Sigma_1 R \leq \Sigma_2$. (The case where $\Sigma_1 L \leq \Sigma_2$ is similar.) Let

$$\begin{aligned}\sigma_1 &= \pi_1[\forall\alpha_1.(\tau_1 \rightarrow \pi_2[\forall\alpha_2.\tau_2])] \\ \sigma_2 &= \pi_1[(\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\pi_2[\forall\alpha_2.\tau_2])] \\ \sigma_3 &= \pi_1[(\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\pi_2[S_2(\tau_2)])]\end{aligned}$$

for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 and τ_2 , type variables α_1 and α_2 , and substitution S_2 . Let $\sigma_4 = \pi_1[\forall\alpha_1.(\tau_1 \rightarrow \pi_2[S_2(\tau_2)])]$. It is immediate that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.

(b) ($\Sigma_1 \geq \Sigma_2$) Let

$$\begin{aligned}\sigma_1 &= \pi_2[\forall\alpha_2.\pi_1[\forall\alpha_1.(\tau_1 \rightarrow \tau_2)]] \\ \sigma_2 &= \pi_2[\forall\alpha_2.\pi_1[(\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\tau_2)]] \\ \sigma_3 &= \pi_2[S_2(\pi_1[(\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\tau_2)])]\end{aligned}$$

for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 and τ_2 , type variables α_1 and α_2 , and substitution S_2 . Let $\sigma_4 = \pi_2[S_2(\pi_1[\forall\alpha_1.(\tau_1 \rightarrow \tau_2)])]$. It is immediate that $\sigma_1 \boxplus \sigma_4$. We may assume that $\text{BHV}(\pi_1[\forall\alpha_1.[\]]) \cap \text{FTV}(S_2(\alpha_2)) = \emptyset$ and that $\alpha_2 \notin \text{FTV}(S_2(\alpha_2))$. Thus,

$$\begin{aligned}\sigma_4 &= \pi_2[S_2(\pi_1[\forall\alpha_1.(S_2(\tau_1) \rightarrow S_2(\tau_2))])] \\ &\boxplus \pi_2[S_2(\pi_1[(\forall\alpha_1.S_2(\tau_1)) \rightarrow (\forall\alpha_1.S_2(\tau_2))])] = \sigma_3\end{aligned}$$

- (c) $(\Sigma_1 \not\leq \Sigma_2 \not\leq \Sigma_1)$ Easy.
4. R_1 is (h-distr +1) and R_2 is (h-distr +1). By cases on whether $\Sigma_1 = \Sigma_2$, $\Sigma_1 < \Sigma_2$, $\Sigma_1 > \Sigma_2$, or $\Sigma_1 \not\leq \Sigma_2 \not\leq \Sigma_1$.
- (a) $(\Sigma_1 = \Sigma_2)$ Let $\sigma_1 = \pi[\forall\alpha_2.\forall\alpha_1.(\tau_1 \rightarrow \tau_2)]$, $\sigma_2 = \pi[\forall\alpha_2.((\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\tau_2))]$, and $\sigma_3 = \pi[(\forall\alpha_2.\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_2.\forall\alpha_1.\tau_2)]$ for appropriate type context $\pi[\]$, types τ_1 and τ_2 , and type variables α_1 and α_2 . Let $\sigma_4 = \pi[\forall\alpha_1.((\forall\alpha_2.\tau_1) \rightarrow (\forall\alpha_2.\tau_2))]$. Since $\forall\alpha_1.\forall\alpha_2.\tau = \forall\alpha_2.\forall\alpha_1.\tau$ for any τ , it is easy to see that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.
- (b) $(\Sigma_1 < \Sigma_2)$ Assume that $\Sigma_1 R \leq \Sigma_2$. (The case where $\Sigma_1 L \leq \Sigma_2$ is similar.) Let
- $$\begin{aligned}\sigma_1 &= \pi_1[\forall\alpha_1.(\tau_1 \rightarrow \pi_2[\forall\alpha_2.(\tau_2 \rightarrow \tau_3)])] \\ \sigma_2 &= \pi_1[(\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\pi_2[\forall\alpha_2.(\tau_2 \rightarrow \tau_3)])] \\ \sigma_3 &= \pi_1[(\forall\alpha_1.\tau_1) \rightarrow (\forall\alpha_1.\pi_2[(\forall\alpha_2.\tau_2) \rightarrow (\forall\alpha_2.\tau_3)])]\end{aligned}$$
- for appropriate type contexts $\pi_1[\]$ and $\pi_2[\]$, types τ_1 , τ_2 and τ_3 , and type variables α_1 and α_2 . Let $\sigma_4 = \pi_1[\forall\alpha_1.(\tau_1 \rightarrow \pi_2[(\forall\alpha_2.\tau_2) \rightarrow (\forall\alpha_2.\tau_3)])]$. It is immediate that $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.
- (c) $(\Sigma_1 > \Sigma_2)$ Assume that $\Sigma_1 \geq \Sigma_2 R$. (The case where $\Sigma_1 \geq \Sigma_2 L$ is similar.) Define $\hat{\sigma}_i$ to be σ_i from the previous case with α_1 and α_2 switched, where appropriate values of $\pi_1[\]$, $\pi_2[\]$, τ_1 , τ_2 , and τ_3 are used. Let $\sigma_1 = \hat{\sigma}_1$, $\sigma_2 = \hat{\sigma}_4$, and $\sigma_3 = \hat{\sigma}_3$. Then defining $\sigma_4 = \hat{\sigma}_2$ results in $\sigma_1 \boxplus \sigma_4 \boxplus \sigma_3$.
- (d) $(\Sigma_1 \not\leq \Sigma_2 \not\leq \Sigma_1)$ Easy.

■

Lemma 4.6 *If $\forall\vec{\alpha}.(\sigma_L \rightarrow \sigma_R) \boxplus^* \tau_L \rightarrow \tau_R$, then $\sigma_L \rightarrow \sigma_R \boxplus^* \tau_L \rightarrow \tau_R$.*

Proof: The claim is proven by induction on the length of the sequence $\forall\vec{\alpha}.(\sigma_L \rightarrow \sigma_R) = \mu_1 \boxplus \dots \boxplus \mu_n = \tau_L \rightarrow \tau_R$. For the base case when $n = 1$, the hypothesis of the claim is impossible, hence the claim is true. Consider the case when $n > 1$. By cases on the rule used to prove $\mu_1 \boxplus \mu_2$.

1. (h-inst -1) Let $\mu_1 = \forall\vec{\alpha}.\pi[S(\rho)] \boxplus \forall\vec{\alpha}.\pi[\forall\beta.\rho] = \mu_2$ for appropriate type context $\pi[\]$, type ρ , type variable β , and substitution S . Let $\hat{\mu}_1 = \pi[S(\rho)]$ and $\hat{\mu}_2 = \pi[\forall\beta.\rho]$. By induction, $\hat{\mu}_2 \boxplus^* \mu_n$. Clearly, $\hat{\mu}_1 \boxplus \hat{\mu}_2$. Thus, $\sigma_L \rightarrow \sigma_R = \hat{\mu}_1 \boxplus^* \mu_n = \tau_L \rightarrow \tau_R$.
2. (h-distr -1) Let $\mu_1 = \forall\vec{\alpha}.\pi[(\forall\beta.\rho_L) \rightarrow (\forall\beta.\rho_R)] \boxplus \forall\vec{\alpha}.\pi[\forall\beta.(\rho_L \rightarrow \rho_R)] = \mu_2$ for appropriate type context $\pi[\]$, types ρ_L and ρ_R , and type variable β . Let $\hat{\mu}_1 = \pi[(\forall\beta.\rho_L) \rightarrow (\forall\beta.\rho_R)]$ and $\hat{\mu}_2 = \pi[\forall\beta.(\rho_L \rightarrow \rho_R)]$. By induction, $\hat{\mu}_2 \boxplus^* \mu_n$. Clearly, $\hat{\mu}_1 \boxplus \hat{\mu}_2$. Thus, $\sigma_L \rightarrow \sigma_R = \hat{\mu}_1 \boxplus^* \mu_n = \tau_L \rightarrow \tau_R$.

■

All of the preceding lemmas in this section only mention subtyping. Now, those lemmas are used to prove results about typings in $F+\eta$.

Lemma 4.7 *If \mathcal{D} is a derivation ending in $A \vdash MN : \tau$, then there is a derivation \mathcal{D}' ending with the same sequent such that $\text{IDT}(M) \boxplus^* \text{FDT}(M)$ and $\text{IDT}(N) \boxplus^* \text{FDT}(N)$.*

Proof: The structure of derivation \mathcal{D} must look like this:

$$\frac{\frac{A \vdash M : \mu}{A \vdash M : \sigma_2 \rightarrow \tau_2} \text{ (subsum)} \quad \frac{A \vdash N : \sigma}{A \vdash N : \sigma_2} \text{ (subsum)}}{A \vdash MN : \tau_2} \text{ APP} \quad \frac{A \vdash MN : \tau_2}{A \vdash MN : \tau} \text{ (subsum)}$$

By Lemma 4.4, there exists a type μ_1 such that $\mu \boxplus^* \mu_1 \boxplus^* \sigma_2 \rightarrow \tau_2$. If $\mu_1 = \perp$, then $\mu_1 \boxplus^* \sigma_2 \rightarrow \tau_2$, a contradiction. Thus, $\mu_1 = \forall \vec{\alpha}. (\sigma_3 \rightarrow \tau_3)$ for some $\vec{\alpha}$, σ_3 , and τ_3 . By Lemma 4.6, $\sigma_3 \rightarrow \tau_3 \boxplus^* \sigma_2 \rightarrow \tau_2$. Thus,

$$\mu \boxplus^* \sigma_3 \rightarrow \tau_3 \boxplus^* \sigma_2 \rightarrow \tau_2$$

By Lemma ??, it holds that $\sigma_2 \leq \sigma_3$ and $\tau_3 \leq \tau_2$. By Lemma 4.4, since $\sigma \leq \sigma_3$, there exists a type σ_1 such that $\sigma \boxplus^* \sigma_1 \boxplus^* \sigma_3$. Using the (\rightarrow) rule shows that $\sigma_3 \rightarrow \tau_3 \boxplus^* \sigma_1 \rightarrow \tau_3$. (The negative subtyping in $\sigma_1 \boxplus^* \sigma_3$ becomes positive when placed in a negative context.) Thus, let \mathcal{D}' be the following derivation:

$$\frac{\frac{A \vdash M : \mu}{A \vdash M : \sigma_1 \rightarrow \tau_3} \text{ (subsum)} \quad \frac{A \vdash N : \sigma}{A \vdash N : \sigma_1} \text{ (subsum)}}{A \vdash MN : \tau_3} \text{ APP} \quad \frac{A \vdash MN : \tau_3}{A \vdash MN : \tau} \text{ (subsum)}$$

■

Lemma 4.8 *If \mathcal{D} is a derivation ending in $A \vdash MN : \tau$, then there is a derivation \mathcal{D}' ending with the same sequent such that the rewrite steps used in proving $\text{IDT}(M) \sqsubset^* \text{FDT}(M)$ are the following, in this order:*

1. Zero or more uses of (h-inst +1) where the active leaf group is $\langle \varepsilon, P \rangle$ for some P such that $L\Pi \in P$ for some Π . The active leaf group for each use of (h-inst +1) is unchanged from $\text{IDT}(M)$, i.e., earlier uses of (h-inst +1) did not result in additional leaves belonging to P .
2. Zero or more uses of (h-distr +1) with active leaf group $\langle \varepsilon, P \rangle$ for some P .

Proof: By Lemma 4.7, from \mathcal{D} we can construct a derivation \mathcal{D}'' in which $\text{IDT}(M) \boxplus^* \text{FDT}(M)$. We work with \mathcal{D}'' instead of \mathcal{D} . Let $\mu = \text{IDT}(M)$, $\sigma = \text{IDT}(N)$, $\sigma_1 = \text{FDT}(N)$, $\tau_1 = \text{IDT}(MN)$, $\sigma_1 \rightarrow \tau_1 = \text{FDT}(M)$, and define $\vec{\mu}$ so that

$$\mu = \mu_1 \sqsubset \cdots \sqsubset \mu_n = \sigma_1 \rightarrow \tau_1$$

Obviously, $\sigma \leq \sigma_1$ and $\tau_1 \leq \tau$. Assume that $\mu \neq \perp$, because if that is the case, then $\mu \boxplus^* \sigma_1 \rightarrow \tau_1$ by one use of (h-inst +1), which proves the lemma immediately.

For each rewrite step $\mu_i \sqsubset \mu_{i+1}$, let $\langle \Sigma_i, P_i \rangle \in \mu_i$ be its active leaf group and place the rewrite step into one of these categories:

1. Use of (h-inst +1) where $\Sigma_i = \varepsilon$ and $L\Pi \in P_i$ for some Π .
2. Use of (h-distr +1) where $\Sigma_i = \varepsilon$.
3. Use of (h-inst +1) where Σ_i begins with L .

4. Use of (h-distr +1) where Σ_i begins with L .
5. Use of (h-inst +1) or (h-distr +1) where Σ_i begins with R .
6. Use of (h-inst +1) where $\Sigma_i = \varepsilon$ and $L\Pi \notin P_i$ for all Π .
7. Use of (h-inst -1) where Σ_i begins with L .

Initially, category 7 will be empty. We will modify the rewrite sequence to produce a new rewrite sequence in which the rules are used in a more desirable order. As the sequence is modified, the values of μ_i , Σ_i , and P_i are updated for each i to reflect the new sequence, except that no active leaf group is identified for any step in category 7.

First, replace each use of (h-inst +1) in category 6 by a use of (h-distr +1) in category 2 followed by a use of (h-inst +1) in category 5. Since the leaf group involved has no leaves that begin with L , the new (h-distr +1) step merely moves the quantifier to R before the (h-inst +1) step instantiates it. There are now no rewrite steps in category 6.

Second, we will repeatedly swap or alter adjacent rewrite steps until the steps occur in the order in which the categories are listed above. Each iteration of this procedure goes as follows. Pick any i such that $\mu_i \sqsubset \mu_{i+1}$ by a rule use in category j and $\mu_{i+1} \sqsubset \mu_{i+2}$ by a rule use in category k where $j > k$. If $j = 7$, apply Lemma 4.4 to swap the rewrite steps. If $j < 7$ and $\langle \Sigma_{i+1}, P_{i+1} \rangle \in \mu_{i+1}$ is the residual of some leaf group in μ_i other than $\langle \Sigma_i, P_i \rangle$, apply Lemma 4.5 to swap the rewrite steps. Otherwise, one of the following cases applies:

1. ($j = 3, k = 1$) One of the free variables in the instantiation in the first use of (h-inst +1) was captured by a \forall -binding of a fresh variable in the type context at the root. The second use instantiates this binding. Combine the two uses of (h-inst +1) into a single use which goes in category 3.
2. ($j = 3, k = 2$) One of the free variables in the substitution in the use of (h-inst +1) was captured by a \forall -binding of a fresh variable in the type context at the root. The use of (h-distr +1) just moves this binding down to position L , since the binding has no leaves beginning with R . Modify the use of (h-inst +1) to leave the variable free instead of capturing it and replace the use of (h-distr +1) with a use of (h-inst -1) (which goes in category 7) to bind the variable at position L .
3. ($j = 5, k = 2$) The (h-inst +1) rule was used in the first rewrite step. One of the free variables in the instantiation in the use of (h-inst +1) was captured by a \forall -binding of a fresh variable in the type context at the root. The use of (h-distr +1) just moves this binding down to position R , since the binding has no leaves beginning with L . Modify the use of (h-inst +1) so that the variable is captured by a \forall -binding at position R , making the use of (h-distr +1) unnecessary, then delete the use of (h-distr +1).
4. ($j = 4, k = 3$) The use of (h-inst +1) operates on the the result of (h-distr +1). There are two cases:
 - (a) $\Sigma_i R \leq \Pi$ for all $\Pi \in P_i$. Combine the two rewrite steps into a single use of (h-inst +1) which goes in category 3.
 - (b) There is some $\Pi \in P_i$ such that $\Sigma_i L \leq \Pi$. Replace the use of (h-distr +1) by a use of (h-inst -1) which handles all of the leaves which begin with $\Sigma_i L$ followed by a use of (h-distr +1) which handles all of the leaves which begin with $\Sigma_i R$. The new use of (h-inst -1) goes in category 7. Then combine the use of (h-distr +1) and the use of (h-inst +1) as in case 4a.

For $0 \leq i \leq 7$ let π_i be the type in the rewrite sequence which is the result of applying all of the rewrite steps in categories 1 through i . Thus,

$$\mu = \pi_0 \Xi^* \pi_1 \Xi^* \cdots \Xi^* \pi_6 \Xi^* \pi_7 = \sigma_1 \rightarrow \tau_1$$

Suppose there are outermost quantifiers in π_6 . Let $\hat{\pi}_6$ be the result of erasing the outermost quantifiers from π_6 . By Lemma 4.6 and since $\sigma \rightarrow \tau_1$ has no outermost quantifiers, it holds that $\hat{\pi}_6 \Xi^* \pi_7$. Insert uses of (h-inst +1) into the rewrite sequence to go from π_6 to $\hat{\pi}_6$, and then reapply the swapping procedure. No new uses of (h-inst -1) will be generated, so this will not loop.

Since there now are no outermost quantifiers in π_6 , and the rule uses in categories 3 through 3 can not remove outermost quantifiers, this means there must not be any outermost quantifiers in π_2 . Let $\sigma_2 \rightarrow \tau_2 = \pi_2$. By Lemma ??, it holds that $\sigma_1 \leq \sigma_2$ and $\tau_2 \leq \tau_1$.

Suppose that in proving $\mu \Xi^* \pi_1$ there is a use of (h-inst +1) where there is free variable α in the instantiation which is captured by a quantifier at the root. Since this quantifier will itself be instantiated later by another use of (h-inst +1), modify the first use so that it uses the result of the later instantiation instead of α . Repeat this procedure until there are no captures of instantiated variables at the root.

Now we construct the desired derivation. Let \mathcal{D}' be as follows:

$$\frac{\frac{A \vdash M : \mu}{A \vdash M : \sigma_2 \rightarrow \tau_2} \text{ (subsum)} \quad \frac{A \vdash N : \sigma}{A \vdash N : \sigma_2} \text{ (subsum)}}{A \vdash MN : \tau_2} \text{ APP} \\ \frac{A \vdash MN : \tau_2}{A \vdash MN : \tau} \text{ (subsum)}$$

■

5 Reducing Subtyping to Typability

The undecidability of typability is proven by a reduction from the undecidable problem of subtyping. This section develops a method for converting instances of the subtyping problem into instances of typability, using the type checking problem as an intermediate step. The first step is to show the existence of a term which induces an invariant type assumption in its typing derivation. Proceeding from there, we construct larger and larger contexts inducing more and more invariant type assumptions.

Definition 5.1 (Constant Context) A context $C[\]$ is a *constant context* for a set of types $\{\sigma_1, \dots, \sigma_n\}$ if the following conditions hold.

1. $C[\]$ has bound variables x_1, \dots, x_n whose scope includes the hole in $C[\]$.
2. $C[x]$ is typable in $F+\eta$ for any variable x .
3. In every derivation \mathcal{D} in $F+\eta$ for $C[\]$, there is some renaming of free type variables R such that for $1 \leq i \leq n$ it holds that $(\mathcal{G}(\mathcal{D}))(x_i) \leq R(\sigma_i)$.

Lemma 5.2 *There is a context $J[\]$ which is a constant context for $\{\alpha \rightarrow \alpha, \perp\}$.*

Proof: Let the context $N[\ , \]$ with two holes be as follows:

$$\lambda r. \mathbf{let} \ z = (\lambda w. r(w))((wr)(wr))[\] \ \mathbf{in} \\ \mathbf{let} \ y = (\lambda x. (\lambda v. x)(r(x(xr))[\])) \ \mathbf{in} \\ r(yz)(zy)((yr)z)(y(\lambda u. z)) \\ (yy)((yr)(yr))((yrr)(yrr))(y(yr))(y(yrr))$$

Let $N'[\]$ be $N[\]$ with every variable x renamed to x' . Let $J[\]$ be as follows:

$$J[\] \equiv N'[r', N[r(wx')(w(\lambda n.x'))(x'(wrr))(x'(wrrr))], [\]]$$

Then $J[\]$ has the desired properties. First, inspection reveals that $J[\]$ has a bound variable x whose scope includes the hole. Second, $J[\]$ is typable using the following type assumptions:

$$\begin{aligned} x &: \alpha \rightarrow \alpha \\ n, x' &: \beta \rightarrow \beta \\ r, r', v, v' &: \perp \\ w, w', y, y' &: \forall \alpha. ((\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)) \\ u, u', z, z' &: (\forall \alpha. ((\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha))) \rightarrow (\forall \alpha. ((\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha))) \end{aligned}$$

The proof of this is merely tedious and is left to the reader. Third, in every typing of $J[\]$ in $F+\eta$, the type $\alpha \rightarrow \alpha$ is assumed for x for some type variable α . The rest of the proof of the lemma is devoted to showing this fact.

Let \mathcal{D} be any typing of $J[\]$. We wish to show that with respect to \mathcal{D} , it holds that $\mathcal{G}(x) = \alpha \rightarrow \alpha$ for some type variable α . The proof will be in two parts. The first part will be to show that

$$\langle \alpha, \{LL^k, RL^k\} \rangle \prec \mathcal{G}(x) \quad \text{and} \quad \langle \varepsilon, \{LLL^k, LRL^k, RLL^k, RRL^k\} \rangle \prec \mathcal{G}(y)$$

for some $k \geq 0$ and some α . The second, more difficult part will be to show that $k = 0$.

Due to the subterms (yy) and $((yr)(yr))$, it must be that $\text{quantsign}(\mathcal{G}(y), L^\omega) = +1 = \text{quantsign}(\mathcal{G}(y), RL^\omega)$. Thus, it must hold that

$$\text{quantsign}((\lambda y.\square), L^\omega) = -1 = \text{quantsign}((\lambda y.\square), LRL^\omega)$$

Due to the subterm $((\lambda y.\square)(\lambda x.(\lambda v.x)\square))$ it must be the case that

$$\text{quantsign}((\lambda x.(\lambda v.x)\square), L^\omega) = +1 = \text{quantsign}((\lambda x.(\lambda v.x)\square), RL^\omega)$$

This implies that $\text{quantsign}(\mathcal{G}(x), L^\omega) \neq +1$ and that $\text{quantsign}(((\lambda v.x)\square), L^\omega) \neq -1$ which implies that $\text{quantsign}(\mathcal{G}(x), L^\omega) \neq -1$. Thus, $\text{quantsign}(\mathcal{G}(x), L^\omega) = 0$.

Since $\text{quantsign}(\mathcal{G}(x), L^\omega) = 0$ and $\text{quantsign}(((\lambda v.x)\square), L^\omega) \neq -1$, it holds that $\text{quantsign}(((\lambda v.x)\square), L^\omega) = 0$. Thus, $\text{leafvar}(\mathcal{G}(x), L^\omega) = \text{leafvar}(((\lambda v.x)\square), L^\omega)$ and $\text{leaf}(\mathcal{G}(x), L^\omega) = \text{leaf}(((\lambda v.x)\square), L^\omega)$. Since

$$\text{quantsign}((\lambda x.(\lambda v.x)\square), L^\omega) = +1 = \text{quantsign}((\lambda x.(\lambda v.x)\square), RL^\omega)$$

it must be the case that $\langle \varepsilon, \{LL^{k_1}, RL^{k_1}\} \rangle \prec \text{IDT}(\lambda x.(\lambda v.x)\square)$ where $\text{leaf}(\mathcal{G}(x), L^\omega) = L^{k_1}$. By repeated uses of Lemma 3.2, it holds for some k_2 and k_3 where $k \leq k_2 \leq k_3$ that

$$\begin{aligned} \langle \varepsilon, \{LL^{k_2}, RL^{k_2}\} \rangle &\prec \text{FDT}(\lambda x.(\lambda v.x)\square) \\ \langle L, \{LLL^{k_2}, LRL^{k_2}\} \rangle &\prec \text{FDT}(\lambda y.\square) \\ \langle L, \{LLL^{k_3}, LRL^{k_3}\} \rangle &\prec \text{IDT}(\lambda y.\square) \\ \langle \varepsilon, \{LL^{k_3}, RL^{k_3}\} \rangle &\prec \mathcal{G}(y) \end{aligned}$$

Due to the subterms (ww) and $((wr)(wr))$, it must be that $\text{quantsign}(\mathcal{G}(w), L^\omega) = +1 = \text{quantsign}(\mathcal{G}(w), RL^\omega)$. This implies that

$$\text{quantsign}((\lambda w.\square), L^\omega) = -1 = \text{quantsign}((\lambda w.\square), LRL^\omega)$$

Due to the subterm $((\lambda z.\square)(\lambda w.\square))$, it must be that $\mathit{quantsign}((\lambda z.\square), L^\omega) = +1 = \mathit{quantsign}((\lambda z.\square), LLRL^\omega)$. Thus, $\mathit{quantsign}(\mathcal{G}(z), LL^\omega) \neq +1 \neq \mathit{quantsign}(\mathcal{G}(z), LRL^\omega)$.

Consider the subterm (zy) . Since

$$\mathit{quantsign}(\mathcal{G}(z), LL^\omega) \neq +1 \neq \mathit{quantsign}(\mathcal{G}(z), LRL^\omega)$$

it must hold that $\mathit{quantsign}(\text{FDT}(y), L^\omega) \neq -1$ and $\mathit{quantsign}(\text{FDT}(y), RL^\omega) \neq -1$. Thus, by Lemma 3.2 for $k_4 \geq k_3$ either $\langle \varepsilon, \{LL^{k_4}, RL^{k_4}\} \rangle \triangleleft \text{FDT}(y)$ or $\langle \beta, \{LL^{k_4}, RL^{k_4}\} \rangle \triangleleft \text{FDT}(y)$ for some type variable β . Thus, one of these two statements must hold:

$$\langle L, \{LLL^{k_4}, LRL^{k_4}\} \rangle \triangleleft \text{FDT}(z) \quad (13)$$

$$\langle \beta, \{LLL^{k_4}, LRL^{k_4}\} \rangle \triangleleft \text{FDT}(z) \quad (14)$$

If (13) holds, then by Lemma 3.2 one of these two statements must hold:

$$\langle L, \{LLL^{k_5}, LRL^{k_5}\} \rangle \triangleleft \mathcal{G}(z) \quad (15)$$

$$\langle \beta, \{LLL^{k_5}, LRL^{k_5}\} \rangle \triangleleft \mathcal{G}(z) \quad (16)$$

If instead (14) is the case, then it would hold that

$$\mathit{quantsign}(\mathcal{G}(z), LL^\omega) = 0 = \mathit{quantsign}(\mathcal{G}(z), LRL^\omega) = 0$$

This would imply that (16) holds where $k_5 = k_4$. Suppose (16) were the case. Then since

$$\mathit{quantsign}((\lambda z.\square), L^\omega) = +1 = \mathit{quantsign}((\lambda z.\square), LLRL^\omega)$$

it would hold that $\langle \varepsilon, \{LLLL^{k_5}, LLRL^{k_5}\} \rangle \triangleleft \text{IDT}(\lambda z.\square)$. By Lemma 3.2 it would have to be true that $\langle \varepsilon, \{LLLL^{k_6}, LLRL^{k_6}\} \rangle \triangleleft \text{FDT}(\lambda z.\square)$ where $k_6 \geq k_5$. However, then the subterm $((\lambda z.\square)(\lambda w.\square))$ would be untypable, since the quantifier can not be at the root. Thus, (15) must be true.

Consider the subterm $(y(\lambda u.z))$. Since $\mathit{quantsign}(\mathcal{G}(z), L^\omega) = -1$, it is easy to see that $\mathit{quantsign}(\text{FDT}(\lambda u.z), RL^\omega) = -1$. Thus, $\mathit{quantsign}(\text{FDT}(y), LRL^\omega) = +1$ implying $\mathit{quantsign}(\mathcal{G}(y), LRL^\omega) = +1$. Due to the subterm $((yrr)(yrr))$, it must hold that $\mathit{quantsign}(\mathcal{G}(y), RRL^\omega) = +1$. Thus, it holds that

$$\mathit{quantsign}((\lambda y.\square), LLRL^\omega) = -1 = \mathit{quantsign}((\lambda y.\square), LRRL^\omega)$$

which implies that

$$\mathit{quantsign}((\lambda x.(\lambda v.x)\square), LRL^\omega) = +1 = \mathit{quantsign}((\lambda x.(\lambda v.x)\square), RRL^\omega)$$

This implies that $\mathit{quantsign}(\mathcal{G}x, RL^\omega) \neq +1$ and that $\mathit{quantsign}(((\lambda v.x)\square), RL^\omega) \neq -1$ which implies that $\mathit{quantsign}(\mathcal{G}(x), RL^\omega) \neq -1$. Thus, $\mathit{quantsign}(\mathcal{G}(x), RL^\omega) = 0$. Due to the subterm $(x(xr))$, it must hold that $L \setminus \mathit{leaf}(\mathcal{G}(x), L^\omega) = R \setminus \mathit{leaf}(\mathcal{G}(x), RL^\omega)$ and $\mathit{leafvar}(\mathcal{G}(x), L^\omega) = \mathit{leafvar}(\mathcal{G}(x), RL^\omega)$. It is easy to see that $\mathit{height}(\mathcal{G}(x)) > 1$. Let g_i stand for $k_i - 1$. Thus, $\langle \alpha, \{LL^{g_1}, RL^{g_1}\} \rangle \triangleleft \mathcal{G}(x)$ for some α . By Lemma 3.2, it holds that $\langle \alpha, \{LL^{g_1}, RL^{g_1}\} \rangle \triangleleft \text{FDT}((\lambda v.x)\square)$. Thus,

$$\langle \varepsilon, \{LLL^{g_1}, LRL^{g_1}, RLL^{g_1}, RRL^{g_1}\} \rangle \triangleleft \text{IDT}(\lambda x.(\lambda v.x)\square)$$

By repeated uses of Lemma 3.2, it holds that:

$$\begin{aligned} \langle \varepsilon, \{LLL^{g_2}, LRL^{g_2}, RLL^{g_2}, RRL^{g_2}\} \rangle &\triangleleft \text{FDT}(\lambda x.(\lambda v.x)\square) \\ \langle L, \{LLLL^{g_2}, LLRL^{g_2}, LRLL^{g_2}, LRRL^{g_2}\} \rangle &\triangleleft \text{FDT}(\lambda y.\square) \\ \langle L, \{LLLL^{g_3}, LLRL^{g_3}, LRLL^{g_3}, LRRL^{g_3}\} \rangle &\triangleleft \text{IDT}(\lambda y.\square) \\ \langle \varepsilon, \{LLL^{g_3}, LRL^{g_3}, RLL^{g_3}, RRL^{g_3}\} \rangle &\triangleleft \mathcal{G}(y) \end{aligned}$$

What remains to be shown is that $g_3 = g_2 = g_1 = 0$. Proving this involves reasoning about infinite paths which end with R^ω instead of L^ω . We show successively that the values $quantsign(\mathcal{G}(x), R^\omega)$, $quantsign(\mathcal{G}(x), LR^\omega)$, and $quantsign(\mathcal{G}(x), LLR^\omega)$ are all 0. This is used to prove that $quant(\mathcal{G}(y), LLLR^\omega) = \varepsilon$, from which the final result will be shown.

Consider the subterms $(y^{(1)}(yr))$ and $(y^{(2)}(yrr))$. Suppose $quantsign(\mathcal{G}(y), R^\omega) = 0$. Then for some j it holds that:

$$\begin{array}{ll} leaf(\text{FDT}(yr), R^\omega) = R^{j+1} & leaf(\text{FDT}(yrr), R^\omega) = R^j \\ quantsign(\text{FDT}(yr), R^\omega) = 0 & quantsign(\text{FDT}(yrr), R^\omega) = 0 \end{array}$$

Thus, it must be the case that

$$\begin{array}{ll} leaf(\text{FDT}(y^{(1)}), LR^\omega) = LR^{j+1} & leaf(\text{FDT}(y^{(2)}), LR^\omega) = LR^j \\ quantsign(\text{FDT}(y^{(1)}), LR^\omega) = 0 & quantsign(\text{FDT}(y^{(2)}), LR^\omega) = 0 \end{array}$$

This implies $quantsign(\mathcal{G}(y), LR^\omega) = +1$. Thus, we know the following:

$$quantsign(\mathcal{G}(y), R^\omega) = +1 \quad \text{or} \quad quantsign(\mathcal{G}(y), LR^\omega) = +1 \quad (17)$$

$$quantsign(\text{IDT}(\lambda y.\square), LR^\omega) = -1 \quad \text{or} \quad quantsign(\text{IDT}(\lambda y.\square), LLR^\omega) = -1 \quad (18)$$

$$quantsign((\lambda x.\square), R^\omega) = +1 \quad \text{or} \quad quantsign((\lambda x.\square), LR^\omega) = +1 \quad (19)$$

Consider the subterm (yy) . By Lemma 3.7 (3c) both of the following statements are true:

$$quantsign(\mathcal{G}(y), LR^\omega) \neq +1 \Rightarrow leaf(\mathcal{G}(y), R^\omega) \leq L \setminus leaf(\mathcal{G}(y), LR^\omega) \quad (20)$$

$$quantsign(\mathcal{G}(y), LLR^\omega) \neq +1 \Rightarrow leaf(\mathcal{G}(y), LR^\omega) \leq L \setminus leaf(\mathcal{G}(y), LLR^\omega) \quad (21)$$

Let $leaf(\mathcal{G}(x), R^\omega) = R^f$. Suppose $quantsign(\mathcal{G}(x), R^\omega) = +1$. This would imply $leaf(\text{FDT}((\lambda v.x)\square), R^\omega) \geq R^f$ which would imply $R \setminus leaf(\text{IDT}(\lambda x.(\lambda v.x)\square), R^\omega) \geq R^f$. By Lemma 3.7 (5b), it would hold that $quantsign(\mathcal{G}(y), LR^\omega) \neq +1$ and $L \setminus leaf(\mathcal{G}(y), LR^\omega) \leq R^f$. By (17) this would imply $quantsign(\mathcal{G}(y), R^\omega) = +1$, which by Lemma 3.7 (4a) implies that $quantsign((\lambda x.\square), R^\omega) = +1$ and $leaf(\mathcal{G}(y), R^\omega) \geq leaf(\text{IDT}(\lambda x.(\lambda v.x)\square), R^\omega)$. This implies that $L \setminus leaf(\mathcal{G}(y), LR^\omega) \leq R \setminus leaf(\mathcal{G}(y), R^\omega)$, which contradicts (20). Thus, it must hold that $quantsign(\mathcal{G}(x), R^\omega) = 0$. Due to the subterm (xr) it holds that $f \geq 1$. Let $f_1 = f - 1$.

Since $quantsign(\mathcal{G}(x), R^\omega) = 0$, it is easy to deduce that $quantsign((\lambda v.x)\square), R^\omega) = 0$ and $leaf(((\lambda v.x)\square), R^\omega) = RR^{f_1}$. By (19) it holds that:

$$\langle \varepsilon, \{LRR^{f_1}, RRR^{f_1}\} \rangle \triangleleft \text{IDT}(\lambda x.(\lambda v.x)\square)$$

Suppose that the following does not hold for some $f_2 \geq f_1$:

$$\langle \varepsilon, \{LRR^{f_2}, RRR^{f_2}\} \rangle \triangleleft \text{FDT}(\lambda x.(\lambda v.x)\square) \quad (22)$$

By Lemma 3.2 and (19), if (22) is false then at least one of $quantsign(\text{FDT}(\lambda x.\square), R^\omega)$ and $quantsign(\text{FDT}(\lambda x.\square), LR^\omega)$ must be -1 . Since $quantsign(\sigma, R^\omega)$ can never be -1 , it would hold that $quantsign(\text{FDT}(\lambda x.\square), LR^\omega) = -1$. Let $\Pi = L \setminus leaf(\text{FDT}(\lambda x.\square), LR^\omega)$. By Lemma 3.5, $\Pi \leq R \setminus leaf(\text{FDT}(\lambda x.\square), R^\omega)$. By Lemma 3.7 (4b), it would hold that $quantsign(\mathcal{G}(y), LR^\omega) \neq +1$ and $leaf(\mathcal{G}(y), LR^\omega) \leq L\Pi$. By (17), it would hold that $quantsign(\mathcal{G}(y), R^\omega) = +1$. By Lemma 3.7 (4a), it holds that:

$$leaf(\mathcal{G}(y), R^\omega) \geq leaf(\text{FDT}(\lambda x.\square), R^\omega) \geq R\Pi$$

This would imply that $R \setminus \text{leaf}(\mathcal{G}(y), R^\omega) \geq \Pi \geq L \setminus \text{leaf}(\mathcal{G}(y), LR^\omega)$, contradicting (20). Thus, (22) must be true, implying:

$$\langle L, \{LLRR^{f_2}, LRRR^{f_2}\} \rangle \triangleleft \text{FDT}(\lambda y. \square)$$

Suppose the following does not hold for some $f_3 \geq f_2$:

$$\langle L, \{LLRR^{f_3}, LRRR^{f_3}\} \rangle \triangleleft \text{IDT}(\lambda y. \square) \quad (23)$$

By Lemma 3.2 and (18), if (23) is false then one of $\text{quantsign}(\text{IDT}(\lambda y. \square), LR^\omega)$ and $\text{quantsign}(\text{IDT}(\lambda y. \square), LLR^\omega)$ must be +1 and the other -1. By Lemma 3.3, it would hold that $\text{quantsign}(\text{IDT}(\lambda y. \square), LLR^\omega) = +1$ and $\text{quantsign}(\text{IDT}(\lambda y. \square), LR^\omega) = -1$. Thus, $\text{quantsign}(\mathcal{G}(y), LR^\omega) \neq +1$. By Lemma 3.5 it would hold that

$$LL \setminus \text{leaf}(\text{IDT}(\lambda y. \square), LLR^\omega) \leq LR \setminus \text{leaf}(\text{IDT}(\lambda y. \square), LRR^\omega)$$

which is equivalent to $L \setminus \text{leaf}(\mathcal{G}(y), LR^\omega) \leq R \setminus \text{leaf}(\mathcal{G}(y), RR^\omega)$ which contradicts (20). Thus, (23) must be true, implying:

$$\langle \varepsilon, \{LRR^{f_3}, RRR^{f_3}\} \rangle \triangleleft \mathcal{G}(y) \quad (24)$$

Let $\Pi = L \setminus \text{leaf}(\mathcal{G}(x), LR^\omega)$. Due to the subterm $(x(xr))$ and since we already know that $\text{quantsign}(\mathcal{G}(x), R^\omega) = 0$, it must hold that $\text{quantsign}(\mathcal{G}(x), LR^\omega) \neq -1$ and that $\Pi \leq R \setminus \text{leaf}(\mathcal{G}(x), R^\omega)$. Suppose $\text{quantsign}(\mathcal{G}(x), LR^\omega) = +1$. By Lemma 3.7 (5b) it would hold that $\text{quantsign}(\mathcal{G}(y), LLR^\omega) \neq +1$ and $LL \setminus \text{leaf}(\mathcal{G}(y), LLR^\omega) \leq \Pi$. Since we already know that $\text{quantsign}(\mathcal{G}(y), LR^\omega) = +1$ and by Lemma 3.7 (5a) it must be the case that $L \setminus \text{leaf}(\mathcal{G}(y), LR^\omega) \geq \text{leaf}(\mathcal{G}(x), R^\omega)$. Thus, it would hold that

$$LR \setminus \text{leaf}(\mathcal{G}(y), LR^\omega) \geq \Pi \geq LL \setminus \text{leaf}(\mathcal{G}(y), LLR^\omega)$$

which contradicts (21). Thus, $\text{quantsign}(\mathcal{G}(x), LR^\omega) = 0$. Due to the subterm $(x(xr))$ it must be the case for some type variable β that:

$$\langle \beta, \{LR^{f_1}, RR^{f_1}\} \rangle \triangleleft \mathcal{G}(x)$$

Of course, everything that has been proven so far about the type $\mathcal{G}(x)$ also applies to $\mathcal{G}(x')$, since the structure of $N'[\quad, \quad]$ is the same as $N[\quad, \quad]$. Consider now the subterms $(w^{(1)}x')$ and $(w^{(2)}(\lambda n.x'))$. Clearly $\text{quantsign}(\mathcal{G}(w), LR^\omega) \neq -1$ since $\text{quantsign}(x', R^\omega) \neq +1$ and $\text{quantsign}((\lambda n.x'), R^\omega) \neq +1$. Suppose it were true that $\text{quantsign}(\mathcal{G}(w), LR^\omega) = 0$. Then $\text{quantsign}(\text{FDT}(w^{(1)}), LR^\omega)$, $\text{quantsign}(\text{FDT}(w^{(2)}), LR^\omega)$, $\text{quantsign}(x', R^\omega)$, and $\text{quantsign}((\lambda n.x'), R^\omega)$ would all be 0. Then it would have to be true that

$$\text{leaf}(\mathcal{G}(x'), R^\omega) = L \setminus \text{leaf}(\mathcal{G}(w), LR^\omega) = R \cdot \text{leaf}(\mathcal{G}(x'), R^\omega)$$

which is a contradiction. Thus, we know that $\text{quantsign}(\mathcal{G}(w), LR^\omega) = +1$.

Due to $((\lambda z. \square)(\lambda w. \square))$, by Lemma 3.7 (5) it holds that $\text{quantsign}(\mathcal{G}(z), LLR^\omega) \neq +1$. By (15) it holds that $\text{leaf}(\mathcal{G}(z), LLR^\omega) \geq LL$. Suppose $\text{quantsign}(\mathcal{G}(z), LLR^\omega) = 0$. Then for some j_1 it would hold that $\langle \varepsilon, \{LLLLR^{j_1}\} \rangle \triangleleft \text{IDT}(\lambda z. \square)$. In order to type the subterm $((\lambda z. \square)(\lambda w. \square))$, it must hold for some j_2 that $\langle LL, \{LLLLR^{j_2}\} \rangle \triangleleft \text{FDT}(\lambda z. \square)$. By Lemma 3.6 this is impossible. Thus, $\text{quantsign}(\mathcal{G}(z), LLR^\omega) = -1$. Due to the subterm (yz) , it holds that $\text{quantsign}(\mathcal{G}(y), LLLR^\omega) = +1$.

Consider the subterms $(x'^{(1)}(wrr))$ and $(x'^{(2)}(wrrr))$. Suppose $\text{quantsign}(\mathcal{G}(w), R^\omega) = 0$. Then for some j it would hold that $\text{leaf}((wrr), R^\omega) = R^{j+1}$ and $\text{leaf}((wrrr), R^\omega) = R^j$. This

would imply that $\text{leaf}(\text{FDT}(x^{(1)}), LR^\omega) = LR^{j+1}$ and $\text{leaf}(\text{FDT}(x^{(2)}), LR^\omega) = LR^j$. Since $\text{quantsign}(\mathcal{G}(x'), LR^\omega) = 0$, this would imply that $\text{quantsign}(\text{FDT}(x^{(2)}), LR^\omega) = -1$, which would imply that $\text{quantsign}(\text{FDT}(wrrr), R^\omega) = +1$, a contradiction. Thus, $\text{quantsign}(\mathcal{G}(w), R^\omega) = +1$. By Lemma 3.7 (5) this implies that $\text{quantsign}(\mathcal{G}(x), LR^\omega) \neq +1$.

Consider the subterm (zy) . By (24) and Lemma 3.2 for some $f_4 \geq f_3$ one of the following possibilities must hold (for some type variable γ in the latter case):

$$\langle \varepsilon, \{LRR^{f_4}, RRR^{f_4}\} \rangle \triangleleft \text{FDT}(y) \quad (25)$$

$$\langle \gamma, \{LRR^{f_4}, RRR^{f_4}\} \rangle \triangleleft \text{FDT}(y) \quad (26)$$

The case of (26) is impossible since we already know that $\text{quantsign}(\mathcal{G}(z), LLR^\omega) = -1$. Thus (25) must be true, implying this:

$$\langle \varepsilon, \{LRR^{f_4}, RRR^{f_4}\} \rangle \triangleleft \text{FDT}(y)$$

$$\langle L, \{LLRR^{f_4}, LRRR^{f_4}\} \rangle \triangleleft \text{FDT}(z)$$

By Lemma 3.2 again for some $f_5 \geq f_4$ it holds that:

$$\langle L, \{LLRR^{f_5}, LRRR^{f_5}\} \rangle \triangleleft \mathcal{G}(z)$$

Consider the subterm $((yr)z)$. Since $\text{quantsign}(\mathcal{G}(z), LR^\omega) = -1$ it must hold that $\text{quantsign}((yr), LLR^\omega) = +1$ which implies that $\text{quantsign}(\mathcal{G}(y), RLLR^\omega) = +1$. By Lemma 3.7 (4) this implies that $\text{quantsign}((\lambda x.(\lambda v.x)\square), RLLR^\omega) = +1$ which implies that $\text{quantsign}(((\lambda v.x)\square), LLR^\omega) \neq -1$ which implies that $\text{quantsign}(\mathcal{G}(x), LLR^\omega) \neq -1$. Since we already know that $\text{quantsign}(\mathcal{G}(y), LLLR^\omega) = +1$ by Lemma 3.7 (5) it must hold that $\text{quantsign}(\mathcal{G}(x), LLR^\omega) = 0$. Thus, $\langle \varepsilon, \{LLLLR^\omega, RLLR^\omega\} \rangle \triangleleft \text{IDT}(\lambda x.\square)$. Repeated uses of Lemma 3.2 show that:

$$\langle \varepsilon, \{LLLLR^\omega, RLLR^\omega\} \rangle \triangleleft \text{FDT}(\lambda x.\square)$$

$$\langle L, \{LLLLR^\omega, LRLLR^\omega\} \rangle \triangleleft \text{FDT}(\lambda y.\square)$$

$$\langle L, \{LLLLR^\omega, LRLLR^\omega\} \rangle \triangleleft \text{IDT}(\lambda y.\square)$$

$$\langle \varepsilon, \{LLLLR^\omega, RLLR^\omega\} \rangle \triangleleft \mathcal{G}(y)$$

Suppose $\text{leaf}(\mathcal{G}(y), LLLR^\omega) \geq LLL$. Consider the subterm (yz) . To type this application, it must hold that $\text{quant}(\text{FDT}(y), LLLR^\omega) = LL$. By Lemma 3.6 this is impossible. Thus, $\text{leaf}(\mathcal{G}(y), LLLR^\omega) \leq LL$. This implies that $g_3 = g_2 = g_1 = 0$, finishing the proof of the lemma. ■

Recall the definitions of \mathbb{B} , \mathbb{U} , and $\mathbb{U}(k)$ from Definition ??.

Lemma 5.3 *There is a constant context for any set $\mathbb{X} \subset \mathbb{B}$.*

Proof: Let $\{\alpha_1, \dots, \alpha_n\} = \text{FTV}(\mathbb{X})$. Let $J_i[\]$ be $J[\]$ from Lemma 5.2 with every variable name x subscripted with i , as in x_i . Let $L_i[\] \equiv ((\lambda q_i.[\])(x_i r_i))$. Then define $H[\]$ as follows.

$$H[\] \equiv J_1[\dots[J_n[L_1[\dots[L_n[\]]]]]]$$

The context $H[\]$ is typable using the type assumption $q_i:\alpha_i$ where the type assumed for x_i is $\alpha_i \rightarrow \alpha_i$.

Now it will be shown that $H[\]$ enforces the desired type assignment. Let \mathcal{D} be an arbitrarily chosen typing of $H[\]$. It is obvious that for $1 \leq i \leq n$ that $\mathcal{G}(x_i) = \alpha_i \rightarrow \alpha_i$ and $\mathcal{G}(q_i) = \alpha_i$ for some α_i . What needs to be shown is that for $i \neq j$ it holds that $\alpha_i \neq \alpha_j$. Assume $i < j$. The typing of the subterm $(\lambda x_j.\square)$ occurs in a position where the type assignment must contain the assumption $x_i:\alpha_i \rightarrow \alpha_i$. If $\alpha_i = \alpha_j$, then it would be impossible to derive the type $\forall \alpha.(\alpha \rightarrow \alpha) \rightarrow (\alpha \rightarrow \alpha)$ for $(\lambda x_j.\square)$. However, in the proof of Lemma 5.2 it is shown that this must happen. Thus, for $i \neq j$ it is the case that $\alpha_i \neq \alpha_j$. ■

The next lemma is an auxiliary lemma. It states that given constants of universal type and some number of constants whose types are type variables, it is possible to construct subterms that behave like constants whose type is partially open with some quantifiers at the root.

Lemma 5.4 *Let $\mu = \alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow \rho$ and $\sigma = \forall \vec{\alpha}.\mu$. Let A be the type assignment such that $A(x) = \sigma$ and $A(y_i) = \beta_i$ for $1 \leq i \leq n$. Let S be the substitution $[\vec{\alpha} := \vec{\beta}]$. The sequent $A \vdash xy_1 \dots y_n : \tau$ is derivable in $F+\eta$ if and only if the sequent $\{z:S(\rho)\} \vdash z : \tau$ is derivable.*

Proof: Each direction of the equivalence is proved separately.

1. (\Leftarrow) Suppose $\{z:S(\rho)\} \vdash z : \tau$ is derivable. Thus, $S(\rho) \leq \tau$ by Lemma ???. It is easy to see that

$$\sigma = \forall \vec{\alpha}.\mu \leq S(\mu) = \beta_1 \rightarrow \dots \rightarrow \beta_n \rightarrow S(\rho)$$

Thus, it holds that $A \vdash xy_1 \dots y_n : S(\rho)$ is derivable. Since $S(\rho) \leq \tau$, it is the case that $A \vdash xy_1 \dots y_n : \tau$ is derivable.

2. (\Rightarrow) Suppose $A \vdash xy_1 \dots y_n : \tau$ is derivable. Let \mathcal{D} be a derivation of this sequent satisfying the properties of Lemma 4.8. Let j range over $\{0, \dots, n\}$. Let S_j be the substitution $[\alpha_1 := \beta_1, \dots, \alpha_j := \beta_j]$. Let N_j be the λ -term $(xy_1 \dots y_j)$. It is sufficient to show the following claim by induction on j :

$$\text{IDT}(N_j) = \forall \alpha_{j+1}. \dots \forall \alpha_n. (\alpha_{j+1} \rightarrow \dots \rightarrow \alpha_n \rightarrow S_j(\rho))$$

In the final case when $j = n$, this claim implies that $\text{IDT}(xy_1 \dots y_n) = \text{IDT}(N_n) = S_n(\rho) = S(\rho)$. Thus, since $\text{IDT}(xy_1 \dots y_n) \leq \tau$, this claim implies that $\{z:S(\rho)\} \vdash z : \tau$ is derivable.

The base case of the induction where $j = 0$ is immediate, since $\mathcal{G}(x)$ satisfies the requirements. In the induction case, we assume the claim holds for $j < n$ and want to prove it for $j + 1$.

By Lemma 4.8, the first subtyping operation used to go from $\text{IDT}(N_j)$ to $\text{FDT}(N_j)$ is a use of (h-inst +1) that instantiates the quantifier $\forall \alpha_{j+1}$:

$$\begin{aligned} & \forall \alpha_{j+1}. \forall \alpha_{j+2}. \dots \forall \alpha_n. (\alpha_{j+1} \rightarrow \dots \rightarrow \alpha_n \rightarrow S_j(\rho)) \\ & \quad \boxplus (\forall \alpha_{j+2}. \dots \forall \alpha_n. (\alpha_{j+1} \rightarrow \dots \rightarrow \alpha_n \rightarrow S_j(\rho))) [\alpha_{j+1} := \pi_{j+1}] \end{aligned}$$

Suppose the replacement π_{j+1} is anything other than β_{j+1} . Observe that $\text{FDT}(N_j)$ must be of the form $\beta_{j+1} \rightarrow \square$, since $\text{FDT}(y_{j+1}) = \beta_{j+1}$. Thus, π_{j+1} can not contain an arrow, since by Lemma 3.1 (4) contracting this would force the value of $\text{quantsign}(\text{FDT}(N_j), L^\omega)$ to be -1 , which is false. Similarly, $\pi_{j+1} \neq \perp$ since by Lemma 3.1 (2) this also forces $\text{quantsign}(\text{FDT}(N_j), L^\omega) = -1$. If π is a type variable and a type context is used in the (h-inst +1) rule which captures this variable, then the (h-inst +1) step is redundant and can be ignored. Also, π_{j+1} can not be a different variable than β_{j+1} , by Lemma 3.1 1. Thus, by elimination the only possibility is that $\pi_{j+1} = \beta_{j+1}$.

Observe that for any type θ that $(S_j(\theta))[\alpha_{j+1} := \beta_{j+1}] = S_{j+1}(\theta)$. So far we know that the subtyping operations from $\text{IDT}(N_j)$ to $\text{FDT}(N_j)$ look like this:

$$\text{IDT}(N_j) \boxplus \forall \alpha_{j+2}. \dots \forall \alpha_n. (\beta_{j+1} \rightarrow \alpha_{j+2} \dots \rightarrow \alpha_n \rightarrow S_{j+1}(\rho)) \boxplus^* \text{FDT}(N_j)$$

By Lemma 4.8, there are no more uses of (h-inst +1) since there are no positive quantifiers remaining which bind leaves that begin with L . The remaining subtyping operations must be uses of (h-distr +1) which operate on quantifiers that are not located at a path which begins

with R . Since $\text{FDT}(N_j)$ has no outermost quantifiers, all such possible uses of (h-distr +1) must be performed. This implies the following result:

$$\text{FDT}(N_j) = \beta_{j+1} \rightarrow \forall \alpha_{j+2} \cdot \dots \cdot \forall \alpha_n \cdot (\alpha_{j+2} \cdots \rightarrow \alpha_n \rightarrow S_{j+1}(\rho))$$

The APP rule must now be used to show that

$$\text{IDT}(N_{j+1}) = \forall \alpha_{j+2} \cdot \dots \cdot \forall \alpha_n \cdot (\alpha_{j+2} \cdots \rightarrow \alpha_n \rightarrow S_{j+1}(\rho))$$

which is the desired result.

■

The next stage is to make constant contexts for all universal types.

Lemma 5.5 *For any k and q , if $\tau \in \mathbb{U}(k+1)$, and if for every $\vec{\sigma}$ such that $\{\sigma_1, \dots, \sigma_p\} \subset (\mathbb{B} \cup \mathbb{U}(k))$ and $\{\sigma_{p+1}, \dots, \sigma_{p+q}\} \subset \mathbb{U}(k+1)$ there exists a constant context for $\vec{\sigma}$, then there exists a constant context for $\{\vec{\sigma}, \tau\}$.*

Proof: The way the desired context is constructed is the same as in Lemma ???. The proof that the context works correctly is quite different however. Perhaps the most important difference is that the desired type is only enforced modulo bicoercibility.

First, we repeat the essential elements of the definition of the context. Let τ be the following type in $\mathbb{U}(k+1)$:

$$\tau = \forall \cdot (\rho_1 \rightarrow \dots \rightarrow \rho_c \rightarrow \alpha_g)$$

Define $\{\alpha_1, \dots, \alpha_n\}$ where $n \geq 2$ so that $\text{BTV}(\tau) \subseteq \{\alpha_1, \dots, \alpha_n\}$.

Define the following sets of types:

$$\begin{aligned} \mathbb{R} &= \{\rho_1, \dots, \rho_c\} \\ \mathbb{S} &= \{\alpha_1, \dots, \alpha_n\} \\ \mathbb{X} &= \{\pi \mid \exists \rho \in \mathbb{R} \cdot \pi \subseteq \rho\} \\ \mathbb{Y} &= \{\varphi \rightarrow \alpha_1 \mid \exists \pi \cdot (\pi \rightarrow \varphi) \in \mathbb{X}\} \\ \mathbb{Z} &= \{\alpha \rightarrow \alpha \mid \alpha \in \mathbb{S}\} \\ \mathbb{W} &= \mathbb{R} \cup \mathbb{S} \cup \mathbb{X} \cup \mathbb{Y} \cup \mathbb{Z} \end{aligned}$$

Let μ_1, \dots, μ_r be an enumeration of \mathbb{W} possibly (probably) containing duplicates such that $n+c \leq r$ and for $1 \leq j \leq n$ it holds that $\mu_j = \alpha_j$ and $\mu_{n+j} = \alpha_j \rightarrow \alpha_j$ and for $1 \leq i \leq c$ it holds that $\mu_{r-c+i} = \rho_i$.

Now define the type assignment A on the λ -term variables $a, b_1, \dots, b_r, d_1, \dots, d_{p+q}$. First, let $A(a) = \perp$. Then, for each $i \in \{1, \dots, r\}$, define $A(b_i)$ in terms of the type $\mu_i \in \mathbb{W}$ as follows:

$$A(b_i) = \begin{cases} \mu_i & \text{if } \mu_i \in (\mathbb{S} \cup \mathbb{Z}), \\ \forall \cdot (\alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow \mu_i) & \text{otherwise.} \end{cases}$$

For each $i \in \{1, \dots, p+q\}$, define $A(d_i) = \sigma_i$. Let $C[\]$ be a constant context enforcing the type assignment A .

Now we define some pieces of the desired context. Define the function m on natural numbers so that $m(i, h) = (((i + h) - 1) \bmod n) + 1$. For $1 \leq i \leq r$ and $h \in \{0, 1\}$ define the λ -term Q_i^h :

$$Q_i^h \equiv \begin{cases} b_{m(j,h)} & \text{if } \mu_i = \alpha_j, \\ b_{n+m(j,h)} & \text{if } \mu_i = \alpha_j \rightarrow \alpha_j, \\ (b_i b_{m(1,h)} \dots b_{m(n,h)}) & \text{otherwise.} \end{cases}$$

Then, define the set \vec{T} of λ -terms:

$$\vec{T} = \{ (z_e(z_i z_j)) \mid \mu_i = \mu_d \rightarrow \mu_f \text{ and } \mu_e = \mu_f \rightarrow \alpha_1 \}$$

Let T_1, \dots, T_s be an arbitrarily chosen enumeration of \vec{T} . Define the term O as follows:

$$O \equiv (y \overbrace{a \dots a}^{r-c})$$

Now define the context $M[\]$ using the subterms defined above. The variables z_1, \dots, z_r which are free in the subterms T_1, \dots, T_s will be captured by bindings within $M[\]$. The variables b_1, \dots, b_r which are free in the subterms Q_1^0, \dots, Q_r^0 and Q_1^1, \dots, Q_r^1 will be free in $M[\]$.

$$M[\] \equiv \left(\begin{array}{l} (\lambda y. a(Q_{n+g}^0(yQ_1^0 \dots Q_r^0)) \\ \quad (Q_{n+g}^1(yQ_1^1 \dots Q_r^1)) \\ ((\lambda x. a(Q_{n+g}^0(xQ_{r-c+1}^0 \dots Q_r^0)) \\ \quad (Q_{n+g}^1(xQ_{r-c+1}^1 \dots Q_r^1))) \\ O) \\ (\lambda z_1 \dots z_r. (\lambda v. z_g)(aT_1 \dots T_s))) \end{array} \right)$$

The context $C[M[\]]$ is a constant context for the set $\{\vec{\sigma}, \tau\}$. First, it must be verified that $C[M[\]]$ is typable in $F+\eta$. Define the type ν as follows:

$$\nu = \forall. (\mu_1 \rightarrow \dots \rightarrow \mu_r \rightarrow \mu_g)$$

It is easy to check that $A \vdash M[a] : \perp$ is derivable in $F+\eta$ using these type assumptions:

$$\{y:\nu, x:\tau, v:\perp\} \cup \{z_i:\mu_i \mid 1 \leq i \leq r\}$$

In fact, the derivation can be performed entirely within System F. Thus, the context $C[M[\]]$ is typable. Since $C[\]$ is already a constant context for the types $\vec{\sigma}$, what remains is to show that $C[M[\]]$ is also a constant context for the type τ . The rest of this proof is devoted to showing this fact.

Let \mathcal{D} be an arbitrarily chosen typing of $C[M[\]]$. Throughout the rest of this proof, unless otherwise specified, let $i \in \{1, \dots, r\}$, $j \in \{1, \dots, n\}$, $h \in \{0, 1\}$, and $\Pi, \Sigma \in \mathcal{P}^\omega$. The proof will show that $\mathcal{G}(x) \leq \tau$. To reach this result, it will be shown that $\mathcal{G}(y) \leq \nu$.

Define the type μ_i^h as follows:

$$\mu_i^h = \mu_i[\alpha_j_{1 \leq j \leq n} := \alpha_{m(j,h)}]$$

We prove that $\mu_i^h \leq \text{FDT}(Q_i^h)$. Suppose $Q_i^h \equiv b_e$. This occurs either when $1 \leq e \leq n$ and $\mu_i^h = \alpha_e$ or when $n+1 \leq e \leq 2n$ and $\mu_i^h = \alpha_{e-n} \rightarrow \alpha_{e-n}$. In this case, it is immediate that $\mu_i^h = \mathcal{G}(b_e) \leq \text{FDT}(Q_i^h)$. Suppose instead that $Q_i^h \equiv (b_i b_{m(1,h)} \dots b_{m(n,h)})$. In this case,

$\mathcal{G}(b_i) = \forall.(\alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow \mu_i)$ and $\mathcal{G}(b_{m(j,h)}) = \alpha_{m(j,h)}$. Then by Lemma 5.4 it holds that $\mu_i^h = \text{IDT}(b_i b_{m(1,h)} \dots b_{m(n,h)})$, which implies the desired result. The fact that $\mu_i^h \leq \text{FDT}(Q_i^h)$ implies the following:

$$\begin{aligned} \text{quantsign}(\text{FDT}(Q_i^h), \Pi) &\neq +1 \\ \text{leaf}(\text{FDT}(Q_i^h), \Pi) &\leq \text{leaf}(\mu_i, \Pi) \end{aligned} \quad (27)$$

Consider these subterms of $M[\]$:

$$\begin{aligned} (Q_{n+g}^0(yQ_1^0 \dots Q_r^0)), & \quad (Q_{n+g}^0(xQ_{r-c+1}^0 \dots Q_r^0)), \\ (Q_{n+g}^1(yQ_1^1 \dots Q_r^1)), & \quad (Q_{n+g}^1(xQ_{r-c+1}^1 \dots Q_r^1)) \end{aligned} \quad (28)$$

Due to (27) and repeated uses of Lemma 3.7 (3) it holds that $\text{quantsign}(\mathcal{G}(y), \Pi) \neq -1$ and $\text{quantsign}(\mathcal{G}(x), \Pi) \neq -1$. Suppose it were the case for some $f \in \{0, \dots, r-1\}$ that $\text{quantsign}(\mathcal{G}(y), R^f L\Pi) = 0$. Then it would hold that $\text{leafvar}(\text{FDT}(yQ_1^0 \dots Q_f^0), L\Pi) = \text{leafvar}(\text{FDT}(yQ_1^1 \dots Q_f^1), L\Pi)$. From this it can be deduced that $\text{leafvar}(\text{FDT}(Q_{f+1}^0), \Pi) = \text{leafvar}(\text{FDT}(Q_{f+1}^1), \Pi)$, which is impossible. Thus, $\text{quantsign}(\mathcal{G}(y), R^f L\Pi) = +1$. Similar reasoning shows that $\text{quantsign}(\mathcal{G}(y), R^r \Pi) = +1$. Thus, by the above reasoning and similar reasoning for x it holds that

$$\begin{aligned} \text{quantsign}(\mathcal{G}(y), \Pi) &= +1 \\ \text{quantsign}(\mathcal{G}(x), \Pi) &= +1 \end{aligned} \quad (29)$$

Consider these subterms of $M[\]$:

$$\begin{aligned} ((\lambda y.\square)(\lambda z_1. \dots \lambda z_r.\square)) \\ ((\lambda x.\square)O) \end{aligned} \quad (30)$$

By (29) and Lemma 3.7 (4a) it holds that

$$\text{quantsign}((\lambda z_1. \dots \lambda z_r.\square), \Pi) = +1 \quad (31)$$

$$\text{quantsign}(O, \Pi) = +1 \quad (32)$$

By repeated uses of Lemma 3.7 (2) it holds for all $i \in \{1, \dots, r\}$ that that

$$\text{quantsign}((\lambda z_i. \dots \lambda z_r.\square), \Pi) \neq -1 \quad (33)$$

$$\text{quantsign}(\mathcal{G}(z_i), \Pi) \neq +1 \quad (34)$$

Consider again the subterms in (28) and (30). Due to (27), (29), and the subterms in (28) it must hold that $\text{leaf}(\mathcal{G}(y), \Pi) \leq \text{leaf}(\nu, \Pi)$ and $\text{leaf}(\mathcal{G}(x), \Pi) \leq \text{leaf}(\tau, \Pi)$. By (29) and Lemma 3.7 (4a), it is the case that

$$\begin{aligned} \text{leaf}((\lambda z_1. \dots \lambda z_r.\square), \Pi) &\leq \text{leaf}(\nu, \Pi) \\ \text{leaf}(O, \Pi) &\leq \text{leaf}(\tau, \Pi) \end{aligned} \quad (35)$$

By repeated uses of Lemma 3.7 (2) it holds that $\text{leaf}(\mathcal{G}(z_i), \Pi) \leq \text{leaf}(\mu_i, \Pi)$.

Now we show that $\text{quantsign}(\mathcal{G}(z_i), \Pi) \neq -1$ by cases on the size of the types $\bar{\mu} \in \mathbb{W}$. If $\mu_i = \alpha_j$ it is obvious that there is no negative position in $\mathcal{G}(z_i)$ at which to put a quantifier, since $\mathcal{G}(z_i)$ is not larger than μ_i . Otherwise, consider the case where $\mu_i = \mu_d \rightarrow \mu_f$ and assume $\mu_e = \mu_f \rightarrow \alpha_1$. Due to (34), the subterm $(z_e(z_i z_d))$ forces $\text{quantsign}(\mathcal{G}(z_i), \Pi) \neq -1$. Thus,

$$\text{quantsign}(\mathcal{G}(z_i), \Pi) = 0 \quad (36)$$

Knowing this, the same induction as in Lemma ?? proves that $leaves(\mathcal{G}(z_i)) = leaves(\mu_i)$ and for $e, f \in \{1, \dots, r\}$ that if $leafvar(\mu_e, \Pi) = leafvar(\mu_f, \Sigma)$ then $leafvar(\mathcal{G}(z_e), \Pi) = leafvar(\mathcal{G}(z_f), \Sigma)$.

For $1 \leq j \leq n$ define $\beta_j = \mathcal{G}(z_j)$. (Note that it has not yet been proven that $e \neq f$ implies $\beta_e \neq \beta_f$.)

Consider yet again the subterms in (28) and (30). By (35) and repeated use of Lemma 3.7 (2) it can now be shown that $leaves(\lambda z_1 \dots \lambda z_r. \square) = leaves(\nu)$. (Observe that here \square stands for $((\lambda v. z_g) \square')$ and thus $FDT(\square) = \beta_g$.) By Lemma 3.7 (4a), it must hold that $leaves(\mathcal{G}(y)) = leaves(\nu)$. By (32) and repeated uses of Lemma 3.7 (3g) it holds that $leaves(O) = leaves(\tau)$. By Lemma 3.7 (4a), it must hold that $leaves(\mathcal{G}(x)) = leaves(\tau)$. Referring to (29) and (27), it must then be the case that $leaves(FDT(Q_i^h)) = leaves(\mu_i)$. Thus, the following results hold:

$$\begin{aligned} leaves(\mathcal{G}(y)) &= leaves(\nu) \\ leaves(\mathcal{G}(x)) &= leaves(\tau) \end{aligned} \tag{37}$$

To show that $\mathcal{G}(y) \leq \nu$ and $\mathcal{G}(x) \leq \tau$, all that is left is to show that property 4 of Lemma 2.9 is satisfied, since (29) and (37) prove properties 1, 2, and 3. We finish proving this for $\mathcal{G}(y)$ first. Consider how leaf-group patterns propagate through the typing of each of these subterms of $M[\]$ in succession:

$$\begin{aligned} &(\lambda z_r. \square) \\ &\quad \vdots \\ &(\lambda z_2. \dots \lambda z_r. \square) \\ &(\lambda z_1. \lambda z_2. \dots \lambda z_r. \square) \end{aligned}$$

Let $e, f \in \{i, \dots, r\}$ such that $e < f$. Suppose that $leafvar(\mu_e, \Pi) = \alpha_j = leafvar(\mu_f, \Sigma)$. If $j < i \leq e$ then due to (33) it must hold that

$$\langle \beta_j, \{R^{e-i}L\Pi, R^{f-i}L\Sigma\} \rangle \triangleleft FDT(\lambda z_i. \dots \lambda z_r. \square)$$

because the type assumption $z_j:\beta_j$ prevents generalization in the ABS/GEN rule. Recall that $\mu_{j+n} = \alpha_j \rightarrow \alpha_j$. Thus, $\mathcal{G}(z_{j+n}) = \beta_j \rightarrow \beta_j$. If $j < i \leq 2n$ then

$$\langle \beta_j, \{R^{j+n-i}LL, R^{j+n-i}LR, R^{e-i}L\Pi, R^{f-i}L\Sigma\} \rangle \triangleleft FDT(\lambda z_i. \dots \lambda z_r. \square) \tag{38}$$

If $1 \leq i \leq j$ then (38) may hold but since the type assumption $z_j:\beta_j$ has been discharged there is the possibility that

$$\langle \varepsilon, \{R^{j+n-i}LL, R^{j+n-i}LR, R^{e-i}L\Pi, R^{f-i}L\Sigma\} \rangle \triangleleft IDT(\lambda z_i. \dots \lambda z_r. \square) \tag{39}$$

If this is the case, then by (33), (29) and Lemma 3.2 it must hold for some d that

$$\begin{aligned} \langle R^d, \{R^{j+n-1}LL, R^{j+n-1}LR, R^{e-1}L\Pi, R^{f-1}L\Sigma\} \rangle &\triangleleft IDT(\lambda z_1. \dots \lambda z_r. \square) \\ \langle R^d, \{R^{j+n-i}LL, R^{j+n-i}LR, R^{e-1}L\Pi, R^{f-1}L\Sigma\} \rangle &\triangleleft \mathcal{G}(y) \end{aligned}$$

The leaf-group pattern matching (39) must hold for some $i \leq j$. Thus, we know for arbitrary $P \subset \mathcal{P}$ that

$$\langle \varepsilon, P \rangle \triangleleft \nu \Rightarrow \langle R^d, P \rangle \triangleleft \mathcal{G}(y) \quad \text{for some } d$$

(This depends on additional reasoning to handle paths of the form $R^r\Pi$, which we omit since it is similar.)

We have shown that half of property 4 of Lemma 2.9 is satisfied for relating $\mathcal{G}(y)$ with ν . Now we complete the proof that $\mathcal{G}(y) \leq \nu$. Suppose for some d where $leafvar(\mu_e, \Pi) = \alpha_j \neq$

$\alpha_l = \text{leafvar}(\mu_f, \Sigma)$ and $j < l$ that $\langle R^d, \{R^{e-1}L\Pi, R^{f-1}L\Sigma\} \rangle \in \mathcal{G}(y)$. This implies for some d that $\langle R^d, \{R^{j-1}L, R^{l-1}L\} \rangle \in \mathcal{G}(y)$, which prevents the typing of the subterm $(yQ_1^0 \dots Q_j^0 \dots Q_l^0)$ and thus is impossible. We may conclude for all $P \subset \mathcal{P}$ that $\langle R^d, P \rangle \in \mathcal{G}(y)$ for some d implies $\langle \varepsilon, P \rangle \in \nu$. (Once again, this depends on additional reasoning to handle paths of the form $R^r\Pi$.) Therefore, $\mathcal{G}(y) \leq \nu$.

Now we finish proving that $\mathcal{G}(x) \leq \tau$. Let P^+ be the collection of all leaves belonging to leaf groups in τ which contain only positive leaves:

$$P^+ = \{ \Pi \mid \langle \varepsilon, P \rangle \in \tau \text{ and } \Pi \in P \text{ and } \text{sign}(P) = \{+1\} \}$$

Let the sequence P_1^-, \dots, P_t^- be all the leaf sets from leaf groups in τ which contain a negative leaf:

$$\{P_1^-, \dots, P_t^-\} = \{ P \mid \langle \varepsilon, P \rangle \in \tau \text{ and } -1 \in \text{sign}(P) \}$$

It is sufficient to show for $1 \leq i \leq t$ that there exists some d such that $\langle R^d, P_i^- \rangle \in \mathcal{G}(x)$. Any leaf groups in $\mathcal{G}(x)$ formed solely from leaves in P^+ can be ignored, since they do not affect whether $\mathcal{G}(x)$ and τ satisfy property 4 of Lemma 2.9.

First, we show for $1 \leq i \leq t$ that there exists some d such that $\langle R^d, P_i^- \rangle \in \mathcal{G}(x)$. Consider how leaf-group patterns propagate through the typing of each of these subterms of $M[]$ in succession:

$$\begin{array}{c} y \\ (ya) \\ \vdots \\ (y \overbrace{a \cdots a}^{r-c}) \end{array}$$

Observe that for some d it holds that $\langle R^d, R^{r-c} \cdot P_i^- \rangle \in \mathcal{G}(y)$. For $0 \leq f \leq r - c$ it holds by (29), (37), and Lemma 3.7 (3) that

$$\begin{aligned} \text{quantsign}(\text{FDT}(y \overbrace{a \cdots a}^f), \Pi) &= +1 \\ \text{leaves}(\text{FDT}(y \overbrace{a \cdots a}^f)) &= R^f \setminus \text{leaves}(\mathcal{G}(y)) \end{aligned}$$

Since P_i^- contains at least one negative leaf, by repeated uses of Lemma 3.4, for $0 \leq f \leq r - c$ there must exist some d such that

$$\langle R^d, R^{r-c-f} \cdot P_i^- \rangle \in \text{FDT}(y \overbrace{a \cdots a}^f)$$

This implies that there must exist some d such that $\langle R^d, P_i^- \rangle \in \mathcal{G}(x)$, the desired result.

Now, suppose that for $1 \leq i \leq t$ there exists some d and some finite $\Pi \notin P_i^-$ such that $\langle R^d, P_i^- \cup \{\Pi\} \rangle \in \mathcal{G}(x)$. A contradiction will be derived from this supposition. By (27), (29), (37), and Lemma 3.7 (3), it holds for $0 \leq f \leq c$ that

$$\begin{aligned} \text{quantsign}(\text{FDT}(xQ_{r-c+1}^0 \cdots Q_{r-c+f}^0), \Pi) &\neq -1 \\ \text{leaves}(\text{FDT}(xQ_{r-c+1}^0 \cdots Q_{r-c+f}^0)) &= R^f \setminus \text{leaves}(\mathcal{G}(x)) \end{aligned}$$

Pick a Σ and e such that $R^e L \Sigma \in P_i^-$ and $\text{sign}(\Sigma) = +1$. Assume that $\Pi = R^l L \Delta$ and $e \leq l$. (The reasoning when $\Pi = R^c$ or $l < e$ is similar.) By repeated uses of Lemma 3.4, for $0 \leq f < e$ one of

these two cases must hold:

$$\begin{aligned} \langle R^d, R^f \setminus P_i^- \rangle &< \text{FDT}(xQ_{r-c+1}^0 \dots Q_{r-c+f}^0) && \text{for some } d \\ \langle \gamma, R^f \setminus P_i^- \rangle &< \text{FDT}(xQ_{r-c+1}^0 \dots Q_{r-c+f}^0) && \text{for some } \gamma \end{aligned} \quad (40)$$

In order to type the application of $(xQ_{r-c+1}^0 \dots Q_{r-c+e}^0)$ to $Q_{r-c+e+1}^0$, case (40) must hold when $f = e$. This implies that (40) holds when $f > e$. From this we may conclude that $\text{leafvar}(\text{FDT}(Q_{r-c+e+1}^0), \Sigma) = \text{leafvar}(\text{FDT}(Q_{r-c+l+1}^0), \Delta)$, which is a contradiction. Therefore, there can not be any finite $\Pi \notin P_i^-$ such that there exists some d such that $\langle R^d, P_i^- \cup \{\Pi\} \rangle < \mathcal{G}(x)$. Thus, $\mathcal{G}(x) \leq \tau$. ■

A simple induction over the k and q parameters of Lemma 5.5 shows that there is a constant context for every subset of \mathbb{U} .

Lemma 5.6 *There is a constant context for any set $\mathbb{X} \subset \mathbb{U}$.*

Proof: If $P(x, y)$ is a predicate over the natural numbers, then the following is the principle of transfinite induction over ω^2 :

$$\left. \begin{array}{l} P(0, 0) \\ P(x, y) \Rightarrow P(x, y + 1) \\ (\forall y. P(x, y)) \Rightarrow P(x + 1, 0) \end{array} \right\} \Rightarrow \forall x. \forall y. P(x, y)$$

Let $P(x, y)$ be the predicate “there exists a constant context for any set of y types from $\mathbb{U}(x + 1)$ plus any number of types in $\mathbb{B} \cup \mathbb{U}(x)$ ”. Lemma 5.3 proves $P(0, 0)$. Lemma 5.5 proves $P(x, y) \Rightarrow P(x, y + 1)$. The statement $(\forall y. P(x, y)) \Rightarrow P(x + 1, 0)$ is obviously true from the definition of $P(x, y)$. Therefore, by induction it holds that $\forall x. \forall y. P(x, y)$. This is equivalent to the claim of the lemma. ■

The final portion of the reduction requires embedding an instance of SUB into an instance of TC. The next three lemmas provide the tools to do this.

Lemma 5.7 *Let σ and μ be open types such that every variable has at least one positive occurrence, one negative occurrence, and one occurrence at a path beginning with L . In other words, if $\langle \beta, P \rangle$ is a leaf group in σ or τ , then $\text{sign}(P) = \{+1, -1\}$ and $L\Pi \in P$ for some Π . Let $\gamma \notin \text{FTV}(\sigma)$. Then there exists a constant context for the following set:*

$$\{\forall.(\sigma \rightarrow \gamma), \forall\delta.((\forall.\mu) \rightarrow \delta) \rightarrow \delta\}$$

Proof: Let $\alpha_1, \dots, \alpha_n$ be the free type variables of μ . Let $\mu = \mu_L \rightarrow \mu_R$. Define these types:

$$\begin{aligned} \tau &= \forall.(\alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow \delta \rightarrow \mu \rightarrow \delta) \\ \rho &= \forall.(\beta \rightarrow (\beta \rightarrow \delta) \rightarrow \delta) \\ \pi_L &= \forall.(\alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow \mu_L) \\ \pi_R &= \forall.(\alpha_1 \rightarrow \dots \rightarrow \alpha_n \rightarrow \mu_R \rightarrow \alpha_1) \end{aligned}$$

Let $C[\]$ be a constant context enforcing this type assignment:

$$\{a:\perp, d:\forall.(\sigma \rightarrow \gamma), u_L:\pi_L, u_R:\pi_R, x:\tau, y:\forall.\mu, z:\rho\} \cup \{b_i:\alpha_i, c_i:\alpha_i \rightarrow \alpha_i, \mid 1 \leq i \leq n + 1\}$$

By Lemma 5.6, $C[\]$ must exist. Let m be a function on natural numbers such that $m(i, h, k) = ((i + h) - 1) \bmod k + 1$. For $h \in \{0, 1\}$ define the following λ -terms:

$$\begin{aligned} X^h &\equiv x b_{m(1, h, n+1)} \cdots b_{m(n+1, h, n+1)} \\ U_L^h &\equiv u_L b_{m(1, h, n)} \cdots b_{m(n, h, n)} \\ U_R^h &\equiv u_R b_{m(1, h, n)} \cdots b_{m(n, h, n)} \end{aligned}$$

Let $D[\]$ be the following context:

$$D[\] \equiv C[\begin{array}{c} a(c_{m(n+1, 0, n+1)}(vX^0)) \\ (c_{m(n+1, 1, n+1)}(vX^1)) \\ a((\lambda v. a((U_R^0(wU_L^0))) \\ (v(\lambda w. (U_R^1(wU_L^1)))))) \\ [\] \\ (zy) \end{array}]]$$

The context $D[\]$ is already a constant context for the type $\forall.(\sigma \rightarrow \gamma)$. The context $D[\]$ is typable in $F+\eta$ by assuming these types for the bound variables v and w :

$$\begin{aligned} v &: \forall \delta. ((\forall \mu. \mu \rightarrow \delta) \rightarrow \delta) \\ w &: \forall \mu. \end{aligned} \tag{41}$$

What remains to be shown is that $D[\]$ induces the desired invariant type assumption of (41). Let \mathcal{D} be an arbitrary typing in $F+\eta$ for $D[\]$. Throughout the rest of this proof, let Π range over \mathcal{P}^ω and h range over $\{0, 1\}$. The reasoning in this proof is very similar to the reasoning in Lemma 5.2 and Lemma 5.5, so some of the detailed explanations are omitted.

First, by Lemma 5.4 we know the following for $h \in \{0, 1\}$:

$$\begin{aligned} \text{IDT}(X^h) &= (\mu \rightarrow \alpha_{n+1})[\alpha_j \underset{1 \leq j \leq n+1}{:=} \alpha_{m(j, h, n+1)}] \\ \text{IDT}(U_L^h) &= \mu_L[\alpha_j \underset{1 \leq j \leq n}{:=} \alpha_{m(j, h, n)}] \\ \text{IDT}(U_R^h) &= (\mu_R \rightarrow \alpha_1)[\alpha_j \underset{1 \leq j \leq n}{:=} \alpha_{m(j, h, n)}] \end{aligned}$$

Now, consider what can be deduced about the *quantsign* values of various paths in the types of various subterms and bound variables. Due to the subterms $(c_{m(n+1, 0, n+1)}(vX^0))$ and $(c_{m(n+1, 1, n+1)}(vX^1))$, it must be the case that $\text{quantsign}(\mathcal{G}(v), \Pi) = +1$ for all $\Pi \in \mathcal{P}^\omega$. The reasoning is the same as the proof in Lemma 5.5 that $\text{quantsign}(\mathcal{G}(y), \Pi) = +1$. Similarly, due to the subterms $(U_R^0(wU_L^0))$ and $(U_R^1(wU_L^1))$, it holds that $\text{quantsign}(\mathcal{G}(w), \Pi) = +1$. Thus, by Lemma 3.7, it holds that $\text{quantsign}((zy), \Pi) = +1$ and $\text{quantsign}(z, R\Pi) = +1$.

Without loss of generality, assume that the typing of the subterm (zy) obeys the restrictions of Lemma 4.8. This assumption is safe because it does not affect the types assumed for any λ -bound variables and thus does not affect whether $D[\]$ is a constant context for the desired types. It must hold that $\text{FDT}(z) = (\forall \theta. \theta \rightarrow (\forall.((\theta \rightarrow \delta) \rightarrow \delta)))$ and $\text{IDT}(zy) = \forall.((\theta \rightarrow \delta) \rightarrow \delta)$ for some type θ . If $\text{quantsign}(\theta, \Pi) = -1$ for some Π , this would imply that $\text{quantsign}((zy), LL\Pi) = -1$, a contradiction. Thus, $\text{quantsign}(\theta, \Pi) \neq -1$. Since $\forall \theta = \text{FDT}(y)$, this means that $\text{quantsign}(y, \Pi) = +1$.

Now, consider what can be deduced about the length of the leaves along various paths in various types. It must hold that $\text{leaf}(\text{FDT}(y), \Pi) \geq \text{leaf}(\mu, \Pi)$. Thus, $\text{leaf}(\theta, \Pi) \geq \text{leaf}(\mu, \Pi)$, which implies that $LL \setminus \text{leaf}((zy), LL\Pi) \geq \text{leaf}(\mu, \Pi)$. By Lemma 3.7, it is the case that $LL \setminus \text{leaf}(\mathcal{G}(v), LL\Pi) \geq$

$leaf(\mu, \Pi)$. Thus, $leaf(\mathcal{G}(w), \Pi) \geq leaf(\mu, \Pi)$. However, due to the subterms (vX^0) and (vX^1) , it must hold that $LL \setminus leaf(\mathcal{G}(v), LL\Pi) \leq leaf(\mu, \Pi)$. Similarly, due to the subterms $(U_R^0(wU_L^0))$ and $(U_R^1(wU_L^1))$, it holds that $leaf(\mathcal{G}(w), \Pi) \leq leaf(\mu, \Pi)$. Therefore,

$$\begin{aligned} leaves(\mu) &= leaves(\text{FDT}(y)) = leaves(\theta) = LL \setminus leaves(zy) \\ &= LL \setminus leaves(\mathcal{G}(v)) = L \setminus leaves(X^h) = leaves(\mathcal{G}(w)) \end{aligned}$$

The subterm (vX^0) forces $leaf(\mathcal{G}(v), LR\Pi) = LR$ and the subterm $(c_{m(n+1,0,n+1)}(v\Box))$ forces $leaf(\mathcal{G}(v), R\Pi) = R$. Thus, $\mathcal{G}(v)$ has the desired shape.

Now, it is time to show that $\mathcal{G}(v)$ has the correct leaf groups. Suppose $\langle \alpha_i, P \rangle \in \mu$, which implies that $\langle \varepsilon, P \rangle \prec \forall.\mu$. Since $sign(P) = \{+1, -1\}$, by Lemma 3.2, it must hold that $\langle \varepsilon, P \rangle \prec \text{FDT}(y)$. Thus, either $\langle \varepsilon, P \rangle \prec \theta$ or $\langle \alpha, P \rangle \prec \theta$ for some type variable α . In the former case, $\langle LL, LL \cdot P \rangle \prec \text{IDT}(zy)$, while in the latter case, $\langle \varepsilon, LL \cdot P \rangle \prec \text{IDT}(zy)$. By multiple uses of Lemma 3.2 this implies that either $\langle LL, LL \cdot P \rangle \prec \mathcal{G}(v)$ or $\langle \varepsilon, LL \cdot P \rangle \prec \mathcal{G}(v)$. Suppose the latter were the case. In the subterm $(v(\lambda w.\Box))$, it holds that $quantsign(\text{FDT}(v), LL\Pi) = +1$ and $quant(\text{FDT}(v), LL\Pi) \neq \varepsilon$. However, by Lemma 3.2, this is impossible. Hence, it must be the case that $\langle LL, LL \cdot P \rangle \prec \mathcal{G}(v)$, which is the desired result.

Suppose $\langle LL, LL \cdot P \rangle \in \mathcal{G}(v)$. Consider the subterm (vX^0) . Without loss of generality, assume that the typing of this subterm satisfies the restrictions of Lemma 4.7. Thus, $\text{FDT}(X^0) = \text{IDT}(X^0) = \mu \rightarrow \alpha_{n+1}$. Thus, $quantsign(\text{FDT}(X^0), \Pi) = 0$, implying $quantsign(\text{FDT}(v), L\Pi) = 0$. By Lemma 3.2, it must be the case that $\langle \alpha, LL \cdot P \rangle \prec \text{FDT}(v)$ for some α . This implies that $\langle \alpha, L \cdot P \rangle \prec \text{FDT}(X^0)$ which implies that $\langle \alpha, P \rangle \prec \mu$. Therefore,

$$\langle LL, LL \cdot P \rangle \in \mathcal{G}(v) \iff \langle \alpha, P \rangle \in \mu \quad \text{for some } \alpha$$

All that is left is to show that $\langle \varepsilon, \{LR, R\} \rangle \in \mathcal{G}(v)$. Since $\langle \varepsilon, \{LR, R\} \rangle \in \text{FDT}(zy)$, by Lemma 3.2 it holds that $\langle \varepsilon, \{LR, R\} \rangle \prec \mathcal{G}(v)$. Since all other leaves are quantified at LL , only $\{LR, R\}$ can be quantified at ε , which implies the desired result. ■

Lemma 5.8 *Let σ and μ be open types such that every variable has at least one positive occurrence, one negative occurrence, and one occurrence at a path beginning with L . Let $\gamma \notin \text{FTV}(\sigma)$. The subtyping problem is undecidable when restricted to instances of the form $\forall.(\sigma \rightarrow \gamma) \leq \forall\alpha.((\forall.\mu) \rightarrow \alpha)$.*

Proof: Inspecting the reduction from SUP to SUB in Theorem ?? reveals that it generates SUB instances of this form except that some of the type leaves are replaced by \perp . However, these occurrences of \perp are only placeholders and can be replaced by a fresh variable which is bound in the appropriate place. ■

Lemma 5.9 *Let σ and μ be open types such that every variable has at least one positive occurrence, one negative occurrence, and one occurrence at a path beginning with L . Let $\gamma \notin \text{FTV}(\sigma)$. Let A be the type assignment $\{x:\forall\gamma.((\forall.\mu) \rightarrow \gamma) \rightarrow \gamma, y:\forall.(\sigma \rightarrow \gamma)\}$. Then $\forall.(\sigma \rightarrow \gamma) \leq \forall\gamma.((\forall.\mu) \rightarrow \gamma)$ if and only if $A \vdash xy : \tau$ is derivable in $\text{F}+\eta$ for some type τ .*

Proof: Suppose that $\forall.(\sigma \rightarrow \gamma) \leq \forall\gamma.((\forall.\mu) \rightarrow \gamma)$. This immediately implies the following subtyping:

$$\frac{\forall.(\sigma \rightarrow \gamma) \leq \forall\beta.(((\forall.\mu) \rightarrow \gamma)[\gamma:=\beta]) \quad \forall\beta.(\gamma[\gamma:=\beta]) \leq \perp}{\forall\gamma.(((\forall.\mu) \rightarrow \gamma) \rightarrow \gamma) \leq (\forall.(\sigma \rightarrow \gamma)) \rightarrow \perp} (\rightsquigarrow)$$

Thus, we get the following typing:

$$\frac{\frac{A \vdash x : \forall \gamma. ((\forall \mu. \mu \rightarrow \gamma) \rightarrow \gamma)}{A \vdash x : (\forall. (\sigma \rightarrow \gamma)) \rightarrow \perp} \text{(subsum)}}{A \vdash xy : \perp} \frac{A \vdash y : \forall. (\sigma \rightarrow \gamma)}{APP}$$

Suppose that $A \vdash xy : \tau$ is derivable for some type τ . The derivation of this sequent must look like this:

$$\frac{\frac{A \vdash x : \forall \gamma. ((\forall \mu. \mu \rightarrow \gamma) \rightarrow \gamma)}{A \vdash x : \pi \rightarrow \tau'} \text{(subsum)}}{A \vdash xy : \tau'} \frac{\frac{A \vdash y : \forall. (\sigma \rightarrow \gamma)}{A \vdash y : \pi} \text{(subsum)}}{APP}$$

$$\frac{A \vdash xy : \tau'}{A \vdash xy : \tau} \text{(subsum)}$$

where these subtypings hold:

$$\begin{aligned} \forall \gamma. ((\forall \mu. \mu \rightarrow \gamma) \rightarrow \gamma) &\leq \pi \rightarrow \tau' \\ \forall. (\sigma \rightarrow \gamma) &\leq \pi \\ \tau' &\leq \tau \end{aligned}$$

By Lemma 2.2, it holds that $\pi \leq (\forall \mu. \mu \rightarrow \forall \vec{\beta}. S_1(\gamma))$ and $\forall \vec{\beta}. S_1(\gamma) \leq \tau'$ for some substitution S_1 and some type variables $\vec{\beta}$. By (trans) it holds that $\forall. (\sigma \rightarrow \gamma) \leq (\forall \mu. \mu \rightarrow \forall \vec{\beta}. S_1(\gamma))$. By Lemma 2.2, it holds that $\forall \mu. \mu \leq \forall \vec{\delta}. S_2(\sigma)$ and $\forall \vec{\delta}. S_2(\gamma) \leq \forall \vec{\beta}. S_1(\gamma)$ for some substitution S_2 and some type variables $\vec{\delta}$. Let S'_2 be the substitution such that $S'_2(\alpha) = \perp$ if $\alpha = \gamma$ and $S'_2(\alpha) = S_2(\alpha)$ if $\alpha \neq \gamma$. Thus, the following subtyping holds:

$$\frac{\forall \mu. \mu \leq \forall \vec{\delta}. S'_2(\sigma) \quad \forall \vec{\delta}. S'_2(\gamma) \leq \gamma}{\forall. (\sigma \rightarrow \gamma) \leq \forall \gamma. ((\forall \mu. \mu \rightarrow \gamma) \rightarrow \gamma)} (\vec{\supset})$$

■

Lemma 5.10 *An undecidable restriction of the SUB problem is reducible to TYP for F+ η .*

Proof: Let σ and μ be open types such that every variable has at least one positive occurrence, one negative occurrence, and one occurrence at a path beginning with L . Let $\gamma \notin \text{FTV}(\sigma)$. By Lemma 5.8, it is undecidable whether $\forall. (\sigma \rightarrow \gamma) \leq \forall \alpha. ((\forall \mu. \mu \rightarrow \alpha))$. By Lemma 5.9, $\forall. (\sigma \rightarrow \gamma) \leq \forall \alpha. ((\forall \mu. \mu \rightarrow \alpha))$ if and only if

$$\{x : \forall \delta. ((\forall \mu. \mu \rightarrow \delta) \rightarrow \delta), y : \forall. (\sigma \rightarrow \gamma)\} \vdash xy : \perp \quad (42)$$

is derivable in F+ η . By Lemma 5.7, there exists a constant context $D[\]$ such that $D[xy]$ is typable if and only if (42) is derivable. ■

Theorem 5.11 *TYP for F+ η is undecidable.*

Proof: By Lemma 5.10. ■

References

- [Bar84] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics*. North-Holland, revised edition, 1984.
- [Gir72] J.-Y. Girard. *Interprétation Fonctionnelle et Elimination des Coupures de l'Arithmétique d'Ordre Supérieur*. Thèse d'Etat, Université Paris 7, 1972.
- [Jim95] T. Jim. Type inference in system F plus subtyping. Manuscript., Dec. 1995.
- [Lei83] D. Leivant. Polymorphic type inference. In *Conf. Rec. 10th Ann. ACM Symp. Princ. Program. Lang.*, pp. 88–98, Austin, TX, U.S.A., Jan. 24–26, 1983.
- [LMS95] G. Longo, K. Milsted, and S. Soloviev. A logic of subtyping. In *Proc. 10th Ann. IEEE Symp. Logic Comput. Sci.*, pp. 292–299, June 26–29, 1995.
- [Mit88] J. C. Mitchell. Polymorphic type inference and containment. *Inf. Comput.*, 76(2/3):211–249, Feb./Mar. 1988.
- [Mit90] J. C. Mitchell. Polymorphic type inference and containment. In G. Huet, ed., *Logical Foundations of Functional Programming*, chapter 8, pp. 153–193. Addison-Wesley, 1990. An earlier, but nearly identical version is [Mit88].
- [Pie92] B. Pierce. Bounded quantification is undecidable. In *Conf. Rec. 19th Ann. ACM Symp. Princ. Program. Lang.*, pp. 305–315. ACM, 1992. Superseded by [Pie94].
- [Pie94] B. Pierce. Bounded quantification is undecidable. *Inf. Comput.*, 112:131–165, 1994.
- [Rey74] J. C. Reynolds. Towards a theory of type structure. In *Symposium on Programming*, vol. 19 of *LNCS*, pp. 408–425, Paris, France, 1974. Springer-Verlag.
- [Tiu95] J. Tiuryn. Equational axiomatization of bicoercibility for polymorphic types. Technical Report 95-004, Comp. Sci. Dept., Boston Univ., Feb. 1995. URL: <ftp://cs-ftp.bu.edu/techreports/95-004-coercibility.ps.Z>.
- [TU95] J. Tiuryn and P. Urzyczyn. The subtyping problem for second-order types is undecidable. Technical report, Inst. of Informatics, Univ. of Warsaw, Nov. 1995. URL: <ftp://ftp.mimuw.edu.pl/pub/users/urzy/sub-undec.ps.Z>.
- [Wel94] J. B. Wells. Typability and type checking in the second-order λ -calculus are equivalent and undecidable. In *Proc. 9th Ann. IEEE Symp. Logic Comput. Sci.*, July 4–6, 1994.
- [Wel95a] J. B. Wells. Typability is undecidable for F+eta. Technical report, Comp. Sci. Dept., Boston Univ., Dec. 1995.
- [Wel95b] J. B. Wells. The undecidability of Mitchell's subtyping relation. Technical report, Comp. Sci. Dept., Boston Univ., December 1995. URL: <ftp://cs-ftp.bu.edu/pub/jbw/subtyping-undecidable.ps.gz>.