J. Symbolic Computation (1992) **11**, 1–**??**

Unification Under a Mixed Prefix

DALE MILLER

Department of Computer Science

University of Pennsylvania

Philadelphia, PA 19107–6389 USA

(Received 8 July 1992)

Unification problems are identified with conjunctions of equations between simply typed λ-terms where free variables in the equations can be universally or existentially quantified. Two schemes for simplifying quantifier alternation, called *Skolemization* and *raising* (a dual of Skolemization), are presented. In this setting where variables of functional type can be quantified and not all types contain closed terms, the naive generalization of first-order Skolemization has several technical problems that are addressed. The method of searching for pre-unifiers described by Huet is easily extended to the mixed prefix setting, although solving flexible-flexible unification problems is undecidable since types may be empty. Unification problems may have numerous incomparable unifiers. Occasionally, unifiers share common *factors* and several of these are presented. Various optimizations on the general unification search problem are as discussed.

1. Introduction

Most first-order unification algorithms are designed to solve existentially quantified sets of equations such as

$$\exists x_1 \ldots \exists x_n [t_1 = s_1 \wedge \ldots \wedge t_m = s_m].$$

That is, the free variables of the terms $t_1, \ldots, t_m, s_1, \ldots, s_m$ are interpreted as being existentially quantified. Of course, if higher-order types are available, constants within the equations can be universally quantified giving a $\forall\exists$ quantifier prefix. In this paper we consider the more general situation where equations must be solved under a mixed quantifier prefix. That is, we shall consider unification problems to be of the form

$$Q_1 x_1 \ldots Q_n x_n [t_1 = s_1 \wedge \ldots \wedge t_m = s_m],$$

where $Q_1, \ldots, Q_n$ are universal and existential quantifiers. Furthermore, we shall allow the terms $t_1, \ldots, t_m, s_1, \ldots, s_m$ to be simply typed λ-terms and the variables $x_1, \ldots, x_n$ to be of primitive or functional type. Equality between closed λ-terms will be identified with $\beta\eta$-convertibility. Such quantified conjunctions of equations are called *unification problems* and *solutions* for them are certain restricted substitutions for the existentially quantified variables that yield closed equations valid in the $\beta\eta$ theory of equality.

0747–7171/90/000000 + 00 $03.00/0

Below we list several reasons for studying $\beta\eta$-unification of simply typed λ-terms in this way.

Constants and variables are explicitly declared by a prefix. The quantifier prefix can be seen as a declarations of how its bound variables are to be interpreted within the equations. In particular, a universal quantifier declares that its bound variable is to be interpreted as a constant within its scope while an existential quantifier declares that its bound variable is to be interpreted as available for substitution. For example, the prefix $\forall x \exists y \forall z \exists u$ will be used to declare that x and z are constants while y and u are variables within a given unification problem. If the unification problem already contained constants, say c and d, these could be explicitly declared by adding $\forall c \forall d$ to the front of the prefix. In this way, it is generally possible to assume that the logic in which these unification problems are considered contains no nonlogical constants: constants are introduced explicitly by a prefix when they are needed.

Empty types can be studied and used. The explicit information of a prefix makes a convenient setting to study the effect of empty types on unification. For example, checking for the existence of unifiers in this case can be much more complex than in the usual case when all types are assumed to be nonempty.

Substitution terms can range over different signatures. The alternation of quantifiers in a prefix can make distinctions between existential variables that are not captured by type distinctions. Consider again a unification problem with prefix $\forall c \forall d \forall x \exists y \forall z \exists u$. Any substitution term for y can contain c, d, and x, while a substitution term for u may contain these as well as z. Thus, different existentially quantified variables can receive substitution terms built from different signatures.

The intimate relation between λ-abstractions and universal quantifiers can be exploited. An equation between abstractions, say $\lambda x t = \lambda x s$, is provable if and only if the quantified equation $\forall x.t = s$ is provable. Thus, the study of universal quantifiers in prefixes and the study of λ-abstractions in terms can be tied together closely. This connection will be exploited several times in this paper.

Skolemization does not provide a simple solution method. When variables of functional type can be existentially quantified, Skolemization does not provide a simple method of reducing the quantifier alternation in a prefix. There are basically two problems with Skolem functions. Types that were empty prior to introducing a Skolem function can become nonempty afterwards. Furthermore, if λ-abstractions are built without restriction from Skolem functions, solutions to a Skolemized unification problem may be difficult to relate back to solutions of the original problem. In order to *deskolemize* solutions and make Skolemization sound (without using a choice axiom), a restriction on the formation of λ-abstractions over Skolem functions must be made.

When higher types are available, Skolemization has a dual. Skolemization can be seen as taking a constant (a universally quantified variable) in a given unification problem and replacing it with a new constant, the Skolem function, at a higher type. In doing so, the alternation of quantifiers can be, in some sense, simplified. There is a dual operation to this, called *raising*, that takes an existentially quantified variable and replaces it with a new existential variable of higher type in such a way that quantifier alternation is again simplified.

Various computation systems generate unification problems with mixed prefixes. The unification problems considered in this paper are those that arise in the Isabelle theorem prover (Paulson, 1989), in the type inference programs described in (Pfenning, 1988), and in interpreters for the higher-order logic programming language λProlog (Nadathur

& Miller, 1988). In fact many of the definitions and motivation for parts of this paper come directly from having implemented one such interpreter. For example, since many programs in λProlog rely only on second-order unification, the restrictions to second-order used in Sections 11 and 12 have practical consequences. Also in λProlog, the so-called flexible-flexible unification problems may need to be presented as part of an answer substitution: as a result, it is important that the form of these unification problems be simple and intelligible. The material on improper factors in Section 11 can be used to simplify such unification problems.

This paper is organized as follows. In Section 2, unification problems are formalized as certain prenex normal formulas and solvable unification problems are defined as those that are provable in a simple proof system. An equivalent notion of solvability using substitutions for existentially bound variables is presented in Section 3. Some of the problems surrounding the use of Skolemization in this higher-order setting are presented and solved in Section 4. Raising is described and proved sound and complete in Section 5. Section 6 illustrates how Skolemization and raising can be used to process unification problems prior to submitting them to unification processes that do not address quantifier alternation. Huet's pre-unification process for simply typed λ-terms (Huet, 1975) is modified to deal directly with mixed prefixes in Section 7. Pre-unification carries unification problems into a class of *flexible-flexible* problems and Section 8 addresses the solvability of these unification problems. Section 9 describes some unification problems that do not have solutions. A generalization to most general unifiers is presented in Section 10, and Sections 11 and 12 use that generalization on some classes of unification problems. We conclude by describing some related work in Section 13.

2. Unification Problems

Unification problems will be formalized in a logic that is essentially a small sublogic of Church's Simple Theory of Types (Church, 1940). Type expressions in this logic are all the closed first-order terms defined using certain primitive types and one binary infix function symbol $\rightarrow$. When reading type expressions, we associate the infix operator $\rightarrow$ to the right. We shall assume that there is at least one primitive type symbol *meta* that denotes the type of unification problems and as many other primitive types as we find useful. There are no type variables. The symbols τ and δ will be used as syntactic variables ranging over type expressions.

Definition 2.1. *Let τ be the type $\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \tau_0$ where τ_0 is primitive and $n \geq 0$. (By convention, if $n = 0$ then τ is simply the type τ_0.) The types $\tau_1, \ldots, \tau_n$ are the* argument types of τ *while the type τ_0 is the* target type of τ. *The* order of τ *is defined as follows: If τ is primitive then τ has order 0; otherwise, the order of τ is one greater than the maximum order of the argument types of τ.*

We shall assume that there is a denumerably infinite list of variables. Constants are only the logical constants $\top$, $\wedge$, $\overset{\tau}{=}$, $\forall_\tau$, and $\exists_\tau$, where τ ranges over type expressions that do not contain the primitive type *meta*. The type annotations on $=$, $\forall$, and $\exists$ will occasionally be dropped when its actual value is not important or can be inferred from context. Untyped λ-terms can be built up from these variables and constants by using λ-abstraction and application. When reading λ-terms, we associate applications to the left. The notions of subterm and free and bound occurrences of variables are defined as usual. Two λ-terms are considered identical if they differ only in an alphabetic change of bound variables. The notation $[N/x]M$ denotes the substitution of λ-term N for all free occurrences of x in M. Of course, bound variables of M must be systematically changed via α-conversion to avoid variable capture. A term is in β-normal form if it contains no occurrence of a β-redex, that is, a formula of the form $(\lambda x M)N$. Every term t can be associated with a unique (up to α-conversion) β-normal term that is β-convertible to t. We shall assume that the reader is familiar with the basic definitions and properties of $\beta\eta$-conversion (see, for example, Chapters 1 – 7 and 13 of (Hindley & Seldin, 1986)).

In this paper we shall use several forms of equality. As mentioned above, λ-terms are equal if they are alphabetic variants. We shall use no special symbol to represent this equality relation, choosing instead to simply write phrases such as "t is equal to s". When the unadorned symbol $=$ is used, it is either the logical constant $\overset{\tau}{=}$ with its annotation dropped or the mathematical symbol relating objects such as sets or lists: exactly which relation is intended should always be clear from context.

Definition 2.2. *A* signature *is a finite set Γ of pairs, written as $x\colon\tau$, where x is a variable and τ is a type expression, and whenever $x\colon\tau \in \Gamma$ and $x\colon\delta \in \Gamma$, τ and δ are the same type. Signatures are used to assign types to variables. The expression $\Gamma + x\colon\tau$ denotes the set $\Gamma - \{x\colon\delta\} \cup \{x\colon\tau\}$ if Γ assigns type δ to x, or the set $\Gamma \cup \{x\colon\tau\}$ otherwise.*

We now define a provability relation for this logic.

Definition 2.3. *Let $\vdash$ be a provability relation between a signature on the left and either a term or pair $t\colon\tau$ where t is a term and τ is a type expression. The symbol $\vdash^*$ is an abbreviation: $\Gamma \vdash^* P$ means $\Gamma \vdash P\colon meta$ and $\Gamma \vdash P$. Below are the axioms and inference rules for $\vdash$.*

(τ_0) $\Gamma \vdash \top\colon meta$, $\Gamma \vdash \wedge\colon meta \to meta \to meta$, $\Gamma \vdash \overset{\tau}{=}\colon \tau \to \tau \to meta$, $\Gamma \vdash \forall_\tau\colon (\tau \to meta) \to meta$, *and* $\Gamma \vdash \exists_\tau\colon (\tau \to meta) \to meta$.

(τ_1) *If* $x\colon\tau \in \Gamma$ *then* $\Gamma \vdash x\colon\tau$.

(τ_2) *If* $\Gamma \vdash M\colon\tau \to \delta$ *and* $\Gamma \vdash N\colon\tau$ *then* $\Gamma \vdash MN\colon\delta$.

(τ_3) *If* $\Gamma + x\colon\tau \vdash M\colon\delta$ *then* $\Gamma \vdash \lambda x\, M\colon\tau \to \delta$.

($\top$) $\Gamma \vdash \top$.

($=$) *If* $\Gamma \vdash M\colon\tau$, $\Gamma \vdash N\colon\tau$, *and* M *$\beta\eta$-converts to* N *then* $\Gamma \vdash M \overset{\tau}{=} N$.

(ξ) *If* $\Gamma + x\colon\tau \vdash^* N \overset{\delta}{=} M$ *then* $\Gamma \vdash \lambda x\, N \overset{\tau\to\delta}{=} \lambda x\, M$.

($\wedge$) *If* $\Gamma \vdash^* P$ *and* $\Gamma \vdash^* Q$ *then* $\Gamma \vdash P \wedge Q$.

($\exists$) *If* $\Gamma \vdash^* [N/x]P$ *and* $\Gamma \vdash N\colon\tau$ *then* $\Gamma \vdash \exists_\tau x\, P$.

($\forall$) *If* $\Gamma + x\colon\tau \vdash^* P$ *then* $\Gamma \vdash \forall_\tau x\, P$.

A proof of P from Γ *is a list of pairs $\langle\Gamma_1, P_1\rangle, \ldots, \langle\Gamma_n, P_n\rangle$ where $n \geq 1$ and (i) Γ_n is Γ and P is P_n, (ii) for all $i = 1, \ldots, n$, $\Gamma_i \vdash P_i$ is either an axiom (an instance of (τ_0), (τ_1), or ($\top$)) or results from previous pairs by an inference rule. If Γ is empty, we write $\vdash P$ instead of $\Gamma \vdash P$.*

The axioms and rules $(\tau_0), (\tau_1), (\tau_2), (\tau_3)$ are those necessary to impose the simple type discipline on λ-terms. The axiom $(=)$ is, of course, a powerful axiom, capturing all of $\beta\eta$-convertibility in one rule. This rule, however, is decidable for λ-terms that have simple types (Chapter 13, (Hindley & Seldin, 1986)).

Signatures only change in the inference rules ($\forall$) and (ξ). In essence, a constant is discharged during these rules: this is best compared to the discharging of an assumption in the proof of an implication in natural deduction. Signatures are often called *type assignments* elsewhere. We shall prefer the former term since we think of variables occurring to the left of $\vdash$ as actually playing the role of constants with respect to unification.

It is immediate to see that the conclusion of the six rules $(\top)$, $(=)$, (ξ), $(\wedge)$, $(\exists)$, and $(\forall)$ could all be written with $\vdash^*$ substituted for $\vdash$. Hence, if $\Gamma \vdash P$ is provable, either P is of the form $t\colon\tau$ or it is $\top$, an equation, a conjunction, an existential, or a universal formula such that $\Gamma \vdash P\colon meta$. In either case, the free variables of P are assigned some type by Γ.

The following two propositions are simple consequences of this definition of provability.

PROPOSITION 2.4. *Let $\Gamma \vdash t\colon\tau$ and $\Gamma \vdash s\colon\tau$. $\Gamma \vdash t \overset{\tau}{=} s$ if and only if t $\beta\eta$-converts to s.*

PROOF. In the reverse direction, this is simply the inference rule $(=)$. The forward direction is not so immediate since there are two inference rules, namely (ξ) and $(=)$, that can prove an equation. If $\Gamma \vdash t \overset{\tau}{=} s$ then there is a proof of this fact that is built first of typing rules $(\tau_0) - (\tau_3)$, a single instance of $(=)$, and then some number $n \geq 0$ of (ξ) rules. The conclusion of the $(=)$ rule must be of the form

$$\Gamma + x_1\colon\tau_1 + \ldots + x_n\colon\tau_n \vdash t' \overset{\delta}{=} s'$$

where τ is $\tau_1 \to \ldots \tau_n \to \delta$ and t and s are equal up to α-conversion to $\lambda x_1 \ldots \lambda x_n.t'$ and $\lambda x_1 \ldots \lambda x_n.s'$, respectively. Since t' and s' are $\beta\eta$-convertible, so are the terms $\lambda x_1 \ldots \lambda x_n.t'$ and $\lambda x_1 \ldots \lambda x_n.s'$. □

Adding the inference rule (ξ), therefore, does not enrich the equations that can be proved using $\beta\eta$-conversion. This rule is added so that the following proposition will hold.

PROPOSITION 2.5. *Assume that x is not in Γ and let $\Gamma \vdash N\colon\tau \to \delta$ and $\Gamma \vdash M\colon\tau \to \delta$. Then, $\Gamma \vdash \forall_\tau x.Nx \overset{\delta}{=} Mx$ if and only if $\Gamma \vdash N \overset{\tau\to\delta}{=} M$.*

PROOF. Assume $\Gamma \vdash \forall_\tau x.Nx \overset{\delta}{=} Mx$. Then this is proved from $\Gamma + x\colon\tau \vdash Nx \overset{\delta}{=} Mx$, from which we can conclude by (ξ) that $\Gamma \vdash \lambda x Nx \overset{\tau\to\delta}{=} \lambda x Mx$. By Proposition 2.4 and η-conversion, we have $\Gamma \vdash N \overset{\tau\to\delta}{=} M$. If we assume $\Gamma \vdash N \overset{\tau\to\delta}{=} M$ then $\Gamma + x\colon\tau \vdash Nx \overset{\delta}{=} Mx$ since $\beta\eta$-conversion is a congruence relation. Finally, by $(\forall)$, we have $\Gamma \vdash \forall_\tau x.Nx \overset{\delta}{=} Mx$. □

The next proposition follows immediately from the fact that it is decidable to determine if an untyped λ-terms has a simple typing and to determine $\beta\eta$-convertibility for simply typed λ-terms.

PROPOSITION 2.6. *Let Γ be a signature and let P be a λ-term that does not contain any occurrences of $\exists$. It is decidable whether or not $\Gamma \vdash^* P$.*

The following proposition lists several transformations of terms of type *meta* that are useful for several later manipulations. We shall think of such transformation rules as rewriting rules, and as we shall see, no significant property of unification problems are changed by using these rewriting rules.

PROPOSITION 2.7. *Let Γ be a signature and let P be a λ-term such that $\Gamma \vdash P\colon meta$. Let P' be one of the following modifications of P.*

(1) *P' is an alphabetic variant of P.*
(2) *P' is the result of replacing a subterm of the form $\forall_\tau x.N \overset{\delta}{\doteq} M$ with one of the form $\lambda x N \overset{\tau\rightarrow\delta}{\doteq} \lambda x M$, or vice versa.*
(3) *P' is the result of replacing a subterm of the form $\forall_\tau x.N$ with N, or vice versa, provided N does not contain x free and does not contain any existential quantifiers.*
(4) *P' is the result of replacing a subterm of the form $\forall_\tau x.N \wedge \forall_\tau x.M$ with $\forall_\tau x.(N \wedge M)$, or vice versa.*
(5) *P' is the result of replacing a subterm of the form $M \wedge \top$ or $\top \wedge M$ with M.*
(6) *P' is the result of replacing a subterm of the form $\exists_\tau x \exists_\delta y M$ with $\exists_\delta y \exists_\tau x M$ or a subterm of the form $\forall_\tau x \forall_\delta y M$ with $\forall_\delta y \forall_\tau x M$.*
(7) *P is the result of replacing a subterm of the form $t = t$ with $\top$.*

Then, $\Gamma \vdash P'\colon meta$, and $\Gamma \vdash P$ if and only if $\Gamma \vdash P'$.

PROOF. Simple inductions on the length of proofs establish the correctness of all of these rewriting rules. □

DEFINITION 2.8. *A* unification problem *is a closed, β-normal λ-term P such that*

(1) *$\vdash P\colon meta$,*
(2) *P is in prenex normal form; that is, no occurrence of an existential or universal quantifier is in the scope of a conjunction (given condition (1) and the restriction on the types of logical constants, no logical constant can occur in the scope of an equation),*
(3) *if P contains an occurrence of $\wedge$, it contains no occurrences of $\top$ ($\top$ denotes an* empty *conjunction), and*
(4) *equations in P are only between terms of primitive type.*

Conditions (3) and (4) are not genuine restrictions; they are added only for convenience. Given Proposition 2.7, any prefix normal term of type meta *can be rewritten into a terms satisfying these two conditions.*

EXAMPLE 2.9. *Let i be a primitive type. Each of the following are provable:*

$$\exists_{i\rightarrow i} X.X \overset{i\rightarrow i}{\doteq} X \qquad \forall_i y \exists_i X.X \overset{i}{\doteq} X \qquad \forall_i X.X \overset{i}{\doteq} X.$$

The first term is proved by existentially generalizing over the equation $\lambda z\ z \overset{i\rightarrow i}{\doteq} \lambda z\ z$. This term is not a unification problem although it can be rewritten into the unification problem $\exists_{i\rightarrow i} X \forall_i y.Xy \overset{i}{\doteq} Xy$.

The unification problem $\exists_i X.X \overset{i}{\doteq} X$ is not provable from the empty signature. This shows that dropping vacuous universal quantifiers is not generally permitted. Similarly, anti-prenexing rules are not generally valid. For example, while $\forall_i y \exists_i X[y \overset{i}{\doteq} y \wedge X \overset{i}{\doteq} X]$ is provable, $\forall_i y[y \overset{i}{\doteq} y] \wedge \exists_i x[X \overset{i}{\doteq} X]$ is not provable.

As this example shows, the nonprenex normal class of terms forms a class that is interesting on its own right. We shall, however, consider only prenex normal formulas in this paper since they will simplify parts of our presentation. While most of the structure of unification in simple types appears to be illustrated using only prenex normal formulas, the problem of mixing logical inference with unification cannot be addressed only with prenex normal formulas. See (Miller, 1991b) for an example of using nonprenex normal formulas to encode that state of a theorem prover that performs both unification and logical deductions.

Occasionally, we refer to a prefix as being of a form described by a sequence of $\forall$'s and $\exists$'s. For example, an important class of prefixes are of the form $\forall\exists\forall$. Such a prefix has zero or more universal quantifiers, followed by zero or more existential quantifiers, followed by zero or more universal quantifiers. Thus, a prefix that is of the form $\exists\forall$ is also of the form $\forall\exists\forall$. A $\forall$-prefix is either empty or a sequence of just universal quantifiers.

Unification problems can easily be embedded into formulas of Church's Simple Theory of Types (Church, 1940) in the following manner. Let P be a unification problem. While P is technically an untyped λ-term, given the fact that it is β-normal and inferred to have type *meta*, every subterm and bound variables of P can be given a unique type. Let P^* be the simply typed λ-term where all those types are attached to all bound variables. Now let P be some particular unification problem and let $\tau_1, \ldots, \tau_n$ $(n \geq 1)$ be the primitive types used to build types in the prefix of P. Let $\mathcal{T}$ be the formulation of the simple theory of types over primitive types $\tau_1, \ldots, \tau_n$ and where *meta* is identified with Church's type o. Let $\vdash_{\mathcal{T}}$ be provability as in (Church, 1940) except that the axioms of choice, description, and infinity are not assumed. The following theorem can be proved by using the cut-elimination theorem for $\mathcal{T}$ (see (Andrews, 1971), for example).

THEOREM 2.10. *If $x_1, \ldots, x_n$ are distinct variables, then $x_1\colon\tau_1 \ldots, x_n\colon\tau_n \vdash P$ if and only if $\vdash_{\mathcal{T}} P^*$.*

The signature $\{x_1\colon\tau_1 \ldots, x_n\colon\tau_n\}$ above is needed to assure that there exist terms of all types. Given the characterization of $\vdash_{\mathcal{T}}$ using general models (Henkin, 1950; Andrews, 1972), we can conclude from this theorem that $\vdash \forall_{\tau_1} x_1 \ldots \forall_{\tau_n} x_n P$ if and only if P^* is true in all general models.

In the next section we present a characterization of provable unification problems using substitutions.

3. Prefix as Declaration

Besides declaring type information for constants and variables in a unification problem, a prefix also indicates, by the relative positions of variables in the prefix, which constants can appear in substitution terms for which variables. For example, the unification problem $\forall_{i\to i} f \exists_i x \forall_i w[(fw) = x]$, where i is a primitive type, has no solution: trying to instantiate x with (fw) will fail to yield a valid equality since the rules of substitution require that the bound variable w in the prefix be changed to some other variable to avoid variable capture. Thus, the resulting equation would be between (fw) and (fw') where w would be a free variable and w' would be a bound variable. Here, the prefix declares that any substitution term for x may contain f free but may not contain w free. The following definitions help to formalize this constraint on substitutions.

DEFINITION 3.1. *A* quantifier prefix *(prefix for short) is a finite list of distinct variables quantified using typed versions of either universal or existential quantifiers. Let $\mathcal{Q}$ be a* *prefix. The λ-term t is a $\mathcal{Q}$*-term *if t contains no constants other than the logical constants and all the free variables of t are bound in $\mathcal{Q}$. If t is a $\mathcal{Q}$-term we will call the pair $\mathcal{Q}t$ a* prefixed term. *Such a prefixed term is of type δ if $\Gamma \vdash t\colon \delta$, where Γ is the set of pairs $x\colon \tau$ where x is bound by either $\forall_\tau$ or $\exists_\tau$ in $\mathcal{Q}$. Finally, the prefix term $\mathcal{Q}t$ is β-normal if t is β-normal, and two prefixed terms $\mathcal{Q}t_1$ and $\mathcal{Q}t_2$ with the same prefix are β-convertible if t_1 is β-convertible to t_2.*

Given a prefix $\mathcal{Q}$ and an untyped λ-term t, it is decidable whether t is a $\mathcal{Q}$-term of a given type. The type of t, however, is not uniquely determined, in general, from $\mathcal{Q}$. For example, the untyped λ-term $\lambda w.w$ can be given the type $\tau \to \tau$ for any type τ with respect to any prefix $\mathcal{Q}$. However, the type of all subexpressions and bound variables of a β-normal prefixed term are uniquely determined from a type given for the prefixed term. Finally, unification problems can be thought of as being prefixed terms: that is, if P is of the form $\mathcal{Q}\mathcal{D}$ where $\mathcal{Q}$ is a string of quantifiers and $\mathcal{D}$ is a (possibly empty) conjunction of equations, then $\mathcal{Q}\mathcal{D}$ is a β-normal prefixed term of type *meta*.

Definition 3.2. *Let $\mathcal{Q}$ be the prefix $\mathcal{Q}_1 Qx \mathcal{Q}_2$ for prefixes $\mathcal{Q}_1$ and $\mathcal{Q}_2$. If y is bound in $\mathcal{Q}_1$ then y* is to the left of *x in $\mathcal{Q}$. If y is bound in $\mathcal{Q}_2$ then y* is to the right of *x in $\mathcal{Q}$. Often reference to the prefix $\mathcal{Q}$ is dropped when it is obvious which prefix is being considered.*

Next we define what it means to substitute into a prefixed term.

Definition 3.3. *Let $\mathcal{Q}$ be a prefix of the form $\mathcal{Q}_1 \exists_\tau x \mathcal{Q}_2$. A term s is $\mathcal{Q}$*-closed for *x if the free variables of s are universally bound in $\mathcal{Q}_1$ and $\Gamma \vdash s\colon \tau$ where Γ is the signature that associates to universal variables in $\mathcal{Q}_1$ the type at which they are bound in $\mathcal{Q}_1$.*

A finite set θ of pairs is a $\mathcal{Q}$-substitution *if, whenever $(x, s) \in \theta$, x is existentially quantified in $\mathcal{Q}$ and s is β-normal and $\mathcal{Q}$-closed for x. Also, θ must be functional; that is, if (x, s_1) and (x, s_2) are members of θ, then s_1 and s_2 are equal terms. Finally, if $(x, s) \in \theta$ then x is in the* domain *of θ and s is in its* range.

A $\mathcal{Q}$-substitution θ can be considered as the following function on $\mathcal{Q}$-terms. If θ is empty, then $\theta(\mathcal{Q}t) := \mathcal{Q}t$. Otherwise, let $(x, s) \in \theta$, let $\mathcal{Q}$ be $\mathcal{Q}_1 \exists x \mathcal{Q}_2$, and let $\theta' := \theta - \{(x, s)\}$. Set $\theta(\mathcal{Q}t) := \theta'(\mathcal{Q}_1 \mathcal{Q}_2 [s/x]t)$. It is easy to show that since s does not contain any free occurrences of variables in the domain of θ, this definition is independent of the choice of (x, s) from θ. We shall generally write the substitution $\{(x_1, s_1), \ldots, (x_n, s_n)\}$ with the more suggestive notation $[x_1 \mapsto s_1, \ldots, x_n \mapsto s_n]$.

Two $\mathcal{Q}$-substitutions are considered equal it they map the same existentially quantified variable to terms that are η-convertible.

There are prefixes $\mathcal{Q}$ for which no $\mathcal{Q}$-substitution exists.

Definition 3.4. *Let $\mathcal{Q}$ be a prefix of the form $\mathcal{Q}_1 \exists_\tau x \mathcal{Q}_2$. If there is no $\mathcal{Q}$-closed term for x then we say that this occurrence of $\exists_\tau$ in $\mathcal{Q}$ is* empty.

Clearly, a $\mathcal{Q}$-substitution exists if and only if all existential variables of $\mathcal{Q}$ are nonempty. Fortunately, determining whether or not there is a $\mathcal{Q}$-substitution given $\mathcal{Q}$ is decidable since it is equivalent to proving formulas in intuitionistic propositional logic. In particular, a type, say τ, can be thought of as denoting a formula in propositional logic defined by considering all its primitive types as propositional symbols and by considering the type constructor → as implication.

Definition 3.5. *Let Δ be a finite set of types and let τ be a type. The binary relation $\Delta \vdash_I \tau$ defined by the three rules below determines intuitionistic provability for propositional implication logic.*

(1) *If $\tau \in \Delta$ then $\Delta \vdash_I \tau$.*
(2) *If $\{\tau_1\} \cup \Delta \vdash_I \tau_2$ then $\Delta \vdash_I \tau_1 \rightarrow \tau_2$.*
(3) *If $\delta_1 \rightarrow \delta_2 \in \Delta$, $\Delta \vdash_I \delta_1$, and $\{\delta_2\} \cup \Delta \vdash_I \tau$ then $\Delta \vdash_I \tau$.*

This relation is shown to be polynomial-space complete in (Statman, 1979).

The following proposition follows immediately from well known results (Hindley & Seldin, 1986).

Proposition 3.6. *Let $\mathcal{Q}$ be a prefix, let Δ be the set of types attributed to the bound variables in $\mathcal{Q}$, and let τ be a type that does not contain the type symbol meta. $\Delta \vdash_I \tau$ if and only if there is $\mathcal{Q}$-term t of type τ.*

Definition 3.7. *Let P be a unification problem with prefix $\mathcal{Q}$. A $\mathcal{Q}$-substitution θ is a* solution *to P if its domain is the set of all existential variables of $\mathcal{Q}$ and $\vdash \theta P$. Since θP contains no existential quantifiers, determining if θP is provable is decidable (Proposition 2.6).*

Given these definitions, we can now use substitutions to characterize those unification problems that are provable.

Theorem 3.8. *If P is a unification problem, then $\vdash P$ if and only if P has a solution.*

Proof. The proof is a straightforward proof by induction on the length of the prefix of P. □

Thus, given a $\mathcal{Q}$-substitution, it can be decided whether or not it is a solution to P. In general, however, it is undecidable whether or not P has a solution. This was first shown for third-order unification problems independently by Huet (1973a) and Lucchesi (1972) and then later for second-order unification problems by Goldfarb (1981). A variation on Goldfarb's result is presented in Section 8.

One final observation concludes this section. Its proof is simple and omitted.

Proposition 3.9. *The set of solutions to a unification problem does not change when using the rewriting rules in Proposition 2.7. In the case of the first rule using α-conversion, the names of the variables in the corresponding substitutions must be changed to account for the change of bound variables in the prefix.*

The set of solutions for a given unification problem occasionally has a regular structure. For example, a first-order ∀∃-unification problem that has any solution can have all its solutions described as closed instances of a *most general unifier (mgu)*. Recognizing the presence of mgu's is valuable for numerous theoretical and practical considerations. As is well known, however, the unification problems we are considering here do not generally have mgu's. The problem of finding a suitable replacement for mgu's as a characterization of large sets of solutions is addressed in Section 10.

4. Skolemization

Consider the two propositions

$$\forall_\tau x \exists_\delta y\ P \quad \text{and} \quad \exists_{\tau\rightarrow\delta} f \forall_\tau x\ [fx/y]P. \tag{1}$$

If we consider validity in general models, then it is valid that the second formula implies the first. A choice principle is needed to show that the first entails the second. If we dualize the quantifiers in (1), we get the two formulas

$$\exists_\tau x \forall_\delta y\ P \quad \text{and} \quad \forall_{\tau\rightarrow\delta} f \exists_\tau x\ [fx/y]P. \qquad (2).$$

In this case, the forward implication is valid while the reverse implication requires a choice principle. We shall show in Section 5 that when restricted to just the setting of unification problems, one of the formulas in (1) is provable if and only if the other formula in (1) is provable (choice is not needed). A similar relationship holds for the two formulas in (2) only if certain restrictions are made on unification problems. These restrictions are investigated in this section.

The process of replacing the first formula in (2) with the second formula will be called *Skolemization.* Notice that in the theorem proving literature, Skolemization is generally defined with respect to the first pair of formulas above since that literature is often involved with searching for refutations of the negation of proposed theorems. Since we shall be looking for direct proofs and not refutations, Skolemization is properly presented using the formulas in (2).

Skolemization can be described from the syntactic point of view as follows: the scoping of quantifiers in $\exists_\tau x \forall_\delta y\ P$ prohibits the variable y from appearing in the substitution term for x. Replacing the variable y in the first formula of (2) with the term fx will successfully encode this restriction: if t is the substitution term for x, then t cannot contain a subterm occurrence of ft.

For our purposes here, the usual presentation of Skolemization for first-order logic (see, for example, (Andrews, 1986)) is problematic on at least four accounts. First, a formal mechanism for forcing Skolem functions f to be *new* is generally achieved by enriching the logic with a denumerably infinite collection of constants that are not permitted in proposed theorems. These constants only enter the prover via Skolemization. As has already been described, we have a simpler and more elegant method for guaranteeing "newness" using prefixes: the Skolem function must be declared in the prefix of the unification problem and there, as all quantified variables, it must be different from any other variable declared for that problem. This way of writing Skolemization is also appealing since it can be viewed as a left rotation of a universal quantifier over an existential quantifier. This is particularly interesting since a dual operation, that is, a left rotation of an existential quantified variables over a universal quantifier, will also interest us (Section 5).

A second and more serious problem is that Skolem functions should not be used in abstractions as if they are genuine functions. Unrestricted uses of Skolem functions in substitutions can only be justified if choice principles are permitted, but such are not available in the proof system presented in Section 2. Consider the following example. In this section, we shall assume that a and b are two primitive types.

Example 4.1. *The unification problem $\forall_{(a\rightarrow b)\rightarrow a} z \exists_a X \forall_b y\ \top$ has no solution: since there is no intuitionistic proof of a from $(a \rightarrow b) \rightarrow a$, there is no λ-term of type a whose only free variables are of type $(a \rightarrow b) \rightarrow a$. If we Skolemize this formula by moving the bound variable y to the left, we get $\forall_{(a\rightarrow b)\rightarrow a} z \forall_{a\rightarrow b} f \exists_a X\ \top$, which does have the solution $X \mapsto z(\lambda w(fw))$ (which is equal to the substitution $X \mapsto zf$).*

Here, the Skolem function f is used as a first class function and not as the simple syntactic device motivated above. If Skolem terms are used only as such a device, then whenever f has type $\tau \rightarrow \delta$ and there are no $\mathcal{Q}$-terms of type τ, then there should be no $\mathcal{Q}\forall_{\tau\rightarrow\delta} f$-terms containing the function f. A restriction on Skolem functions will be presented later in this section by the introduction of "ranked" universal quantifiers. The aspect of unsoundness demonstrated by this example will be fixed by ranks.

A third problem with Skolemization arises when the solutions to unifications problems before and after Skolemizing are compared. It is most desirable to have a bijection between solutions of a problem and its Skolemized form. Unfortunately, there appears to be no such bijection. A "deskolemizing" mapping can be defined but it will not generally convert all solutions to a Skolemized problem back to solutions of the original problem. Furthermore, when deskolemizing can be used, it may be many-to-one, a fact that is illustrated in the next example.

Example 4.2. *The unification problem $\forall_a z \exists_a X \forall_a y\ \top$ has exactly one solution, namely $X \mapsto z$. The Skolemized form of this problem, namely $\forall_a z \forall_{a\rightarrow a} f \exists_a X\ \top$, has an infinite number of different solutions, namely $X \mapsto z$, $X \mapsto fz$, $X \mapsto f(fz)$,* etc. *The deskolemization process described later will be able to collapse all of these solutions into the solution above by identifying f with the constant-valued z function.*

There seems no easy way to improve Skolemization in order to solve this many-to-one correspondence of solutions. Skolemization adds redundancy to a unification problem.

Our final problem with the use of Skolem functions is the most serious and illustrates an aspect of empty types that is not illustrated by Example 4.1.

Example 4.3. *The unification problem $\forall_a z \forall_{b\rightarrow a} g \exists_a X \forall_b y\ \top$ has exactly one solution, namely $X \mapsto z$. Its Skolemized form, that is, $\forall_a z \forall_{b\rightarrow a} g \forall_{a\rightarrow b} f \exists_a X\ \top$ has an infinite number of solutions; $X \mapsto z$, $X \mapsto g(fz)$, $X \mapsto g(f(g(fz)))$,* etc. *No simple interpretation of the Skolem function f would seem to identify all of these substitution terms since there is no $\forall_a z \forall_{b\rightarrow a} g$-term of type b.*

In this example, the prefix to the left of the existential variable X in the original unification problem does not provide enough variables to build a term of type b. After Skolemization, however, the prefix to the left of X can be used to build a term of type b and that term can be use to build new terms of type a. There seems to be no way to identify the new term of intermediate type b built using f with any term in the original problem.

As we shall see in Theorem 4.11, if the type of the variable that gets Skolemized is nonempty (with respect to the prefix to the left of the existential variable), then all solutions to the Skolemized problem can be deskolemized to solutions of the original unification problem.

As was mentioned above, part of the approach to making Skolem functions sound in this setting is to restrict their occurrences within substitution terms. The notion of *rank* must first be introduced.

Definition 4.4. *The constants $\forall^r_\tau$, where $r \geq 0$ and τ is any type with r or more argument types, are called* ranked universal quantifiers. *A variable bound by $\forall^r_\tau$ is said to have* rank r. *The quantifier $\forall^0_\tau$ is identified with $\forall_\tau$. The logical constant $\forall^r_\tau$ has the same type as the logical constant $\forall_\tau$.*

Ranked universal quantifiers will now be permitted in the prefixes of unification problems. Let $\forall^r_\tau$ quantify the variable x and let τ be of the form $\tau_1 \rightarrow \cdots \rightarrow \tau_n \rightarrow \tau_0$ with τ_0 a primitive type and $r \leq n$. Then, n is the maximum number of arguments that occurrences of x can have, while as we define below, r is the number of arguments that occurrences x *must* have. Several of our definitions regarding prefixes must now be extended. The most fundamental extension is in the following definition: other extensions are more immediate. The motivation for the following definition, however, has already been presented: variables that have a positive rank play the role of Skolem functions: they cannot be permitted to act as genuine functions.

Definition 4.5. *Let $\mathcal{Q}$ be a prefix containing possibly positively ranked universal quantifiers. The λ-term t is a* $\mathcal{Q}$-term *if (i) t contains no nonlogical constants and all the free variables of t are bound in $\mathcal{Q}$, (ii) if t contains a free occurrence of r-ranked universally bound variable then that occurrence must have at least r arguments, called its* necessary arguments, *and (iii) if any variable has a free occurrence in any necessary argument then that free occurrence is also free in t.*

The aggregation of a ranked variable with its necessary arguments is intended to be the name of a single, new object. The condition on free variable occurrences in necessary arguments makes certain that an abstraction is not made into that name.

Example 4.6. *Let $\mathcal{Q}$ be the prefix $\forall^1_{a\rightarrow a} f \forall^1_{a\rightarrow a} g \exists_{a\rightarrow a\rightarrow a} A \forall_a x$. Then $(fx), (f(gx))$, and $\lambda w(Aw(gx))$ are all $\mathcal{Q}$-terms. On the other hand, f, $\lambda w(fw)$, and $\lambda w(A(gx)(fw))$ are not $\mathcal{Q}$-terms.*

Notice that the complications mentioned in Examples 4.2 and 4.3 are not solved if we add ranks to the Skolem functions in them and reinterpreted $\mathcal{Q}$-terms using ranked quantifiers. The proofs of Propositions 4.7 and 4.9 below can be found in (Miller, 1987).

Proposition 4.7. *Let $\mathcal{Q}$ be a prefix and let t be a $\mathcal{Q}$-term of type τ. If t is η-convertible to t' then t' is a $\mathcal{Q}$-term of type τ. If t β-reduces to t' then t' is a $\mathcal{Q}$-term of type τ. In particular, if $\mathcal{Q} = \mathcal{Q}_1 \exists_\delta x \mathcal{Q}_2$ and s is a $\mathcal{Q}_1$-term of type δ, then $[s/x]t$ is a $\mathcal{Q}$-term of type τ.*

The converse of the statement regarding β-reduction is not true: if t' reduces to the $\mathcal{Q}$-term t then t' is not necessarily a $\mathcal{Q}$-term. For example, if $\mathcal{Q}$ is $\forall_a y \forall^1_{a\rightarrow a} u$, then the term y is a $\mathcal{Q}$-term while $(\lambda w\ y)u$ is not.

Definition 4.8. *Let f be a variable, and let s, t, and A be λ-terms. The* deskolemizing *operation D^{ft}_s is defined as follows: $D^{ft}_s A$ is the result of replacing all subterms of A that are $\beta\eta$-convertible to ft by s. Of course, it may be necessary to change bound variable names in A to avoid capturing free variables of s.*

Given that we equate two terms if they are alphabetic variants of each other, the operation D^{ft}_s is technically not well defined. For example, if D^{fx}_y is applied to the two equal terms $\lambda x.a(fx)$ and $\lambda z.a(fz)$, two different answers result, namely $\lambda x.ay$ and $\lambda z.a(fz)$. Fortunately, this cannot happen if f is declared to have a positive rank. For example, we have the following result.

Proposition 4.9. *Let $\mathcal{Q}$ be the prefix $\mathcal{Q}_1\forall^{p+1}_{\tau\rightarrow\delta}f\exists_\tau x\mathcal{Q}_2$, let t be a $\mathcal{Q}_1$-term of type τ, and let A and B be $\mathcal{Q}$-terms. Then all the following are true.*

1. *Any occurrence of y in $D^{ft}_y A$ will have at least p arguments and any free variables in that occurrence's first p arguments are free in $D^{ft}_y A$.*
2. *If A is β-reducible to B then $D^{ft}_y A$ is β-reducible to $D^{ft}_y B$.*
3. *If A is the β-normal form of B, then $D^{ft}_y A$ is the β-normal form of $D^{ft}_y B$.*
4. *If A is η-convertible to B then $D^{ft}_y A$ is η-convertible to $D^{ft}_y B$.*

The term $D^{ft}_y A$ may not be a $\mathcal{Q}_1\exists x\forall^p y\mathcal{Q}_2$-term since it may contain occurrences of f. We can, however, prove the following.

Lemma 4.10. *Let $\mathcal{Q}$ be the prefix $\mathcal{Q}_1\forall^{p+1}_{\tau\rightarrow\delta}f\exists_\tau x\mathcal{Q}_2$, let t be a $\mathcal{Q}$-term of type τ, and let A be a $\mathcal{Q}$-term. Also assume that there exist $\mathcal{Q}_1$-terms of type δ. Then A can be completely "deskolemized" with respect to f in the following fashion. Let $\{t, u_1, \ldots, u_m\}$ ($m \geq 0$) be a set of terms that contains all terms u such that fu is a subterm of A. Let $d_1, \ldots, d_m$ be a list of (not necessarily distinct) $\mathcal{Q}_1$-terms of type δ. The term*

$$A' := D^{fu_1}_{d_1}(\ldots(D^{fu_m}_{d_m}(D^{ft}_y A))\ldots)$$

is a $\mathcal{Q}_1\exists x\forall^p y\mathcal{Q}_2$-term.

Proof. Clearly, A' contains no occurrences of f. Consider an occurrence of the p-ranked variable y in A'. Such an occurrence of y arises from removing an occurrence of ft, which has at least p arguments. Hence, all the occurrences of y in A' have at least p arguments. Finally, applications of the D operator cannot introduce any new bound variable dependencies. Therefore, A' is a $\mathcal{Q}_1\exists x\forall^p y\mathcal{Q}_2$-term. □

The following theorem establishes the completeness for Skolemization. Soundness is only established in the case that the type of the variable being Skolemized is nonempty.

Theorem 4.11. *Let $\mathcal{D}$ be a conjunction of equations such that*

$$\mathcal{Q}_1\exists_\tau x\forall^p_\delta y\mathcal{Q}_2\mathcal{D} \qquad (*)$$

is a unification problem. If this problem has a solution, the unification problem

$$\mathcal{Q}_1\forall^{p+1}_{\tau\rightarrow\delta}f\exists_\tau x\mathcal{Q}_2[fx/y]\mathcal{D} \qquad (**)$$

has a solution. If there exists a $\mathcal{Q}_1$-term of the type δ, the converse is true.

Proof. Let θ be a solution for $(*)$ and let t be θx. Let

$$\theta' := \{(z, [ft/y]s) \mid (z, s) \in \theta\}.$$

It is easy to verify that this substitution is a $\mathcal{Q}_1\forall^{p+1} f \exists x \mathcal{Q}_2$-substitution. Furthermore, if u and v are two terms such that θu and θv are $\beta\eta$-convertible, then $[ft/y]\theta u$ and $[ft/y]\theta v$ are $\beta\eta$-convertible. These last two terms, however, are simply equal to $\theta'([fx/y]u)$ and $\theta'([fx/y]v)$. Hence, θ' is a solution to $(**)$.

Let θ be a $\mathcal{Q}_1\forall^{p+1} f \exists x \mathcal{Q}_2$-substitution that is a solution to $(**)$ and let t be θx. Let $\{t, u_1, \ldots, u_m\}$ $(m \geq 0)$ be a set of terms that contains all terms u such that fu is a subterm of some term in the range of θ. Let $d_1, \ldots, d_m$ be a list of (not necessarily distinct) β-normal $\mathcal{Q}_1$-terms. Finally, let E be the operation $Es := D_{d_1}^{fu_1}(\ldots(D_{d_m}^{fu_m}(D_y^{ft}s))\ldots)$, and let $\theta' := \{(z, Es) \mid (z, s) \in \theta\}$. We now show that θ' is a solution to $(*)$.

Let $(z, s) \in \theta'$. If z is in the prefix $\mathcal{Q}_2$ then s will contain no occurrences of f but may contain occurrences of y. If z is x, then s is Et, which cannot contain any occurrences of y since ft is not a subterm of t. If z is in the prefix $\mathcal{Q}_1$ then Es is the same as s since s contains no occurrences of f. Thus, using Lemma 4.10, θ' is a $\mathcal{Q}_1\exists x \forall^p y \mathcal{Q}_2$-substitution. Furthermore, let u and v be two terms such that $\theta([fx/y]u)$ and $\theta([fx/y]v)$ are $\beta\eta$-convertible. These two terms are also equal to $[ft/y]\theta u$ and $[ft/y]\theta v$, respectively. By repeated use of Proposition 4.9, the two terms $E([ft/y]\theta u)$ and $E([ft/y]\theta v)$ are $\beta\eta$-convertible. But these terms are equal to $E(\theta u)$ and $E(\theta v)$, which are the terms $\theta' u$ and $\theta' v$. Hence, θ' is a solution to $(*)$. □

Definition 4.12. *If P is the unification problem $(*)$ and P' is the unification problem $(**)$ above, we say that P' is the result of* Skolemizing *P at $\forall_\delta^p y$.*

Example 4.13. *The unification problem*

$$\forall_{a\to a} f \forall_{a\to a} g \forall_a u \exists_a X \forall_a y \exists Z[(fX(gy)) \overset{a}{=} (fuZ)]$$

can be Skolemized to the unification problem

$$\forall_{a\to a} f \forall_{a\to a} g \forall_a u \forall_{a\to a}^1 k \exists_a X \exists_a Z[fX(g(kX)) \overset{a}{=} fuZ].$$

This problem has the solution $\{(X, u), (Z, g(ku))\}$ and it can be deskolemized to $\{(X, u), (Z, gy)\}$.

For another example, consider the unification problem

$$\forall_{a\to a\to a} g \forall_a x \exists_{a\to a} F \forall_a y \exists_a Z[Fy \overset{a}{=} gyx \wedge Z \overset{a}{=} Fy].$$

This can be Skolemized to the unification problem

$$\forall_{a\to a\to a} g \forall_a x \forall_{(a\to a)\to a}^1 k \exists_{a\to a} F \exists_a Z[F(kF) \overset{a}{=} g(kF)x \wedge Z \overset{a}{=} F(kF)].$$

Here, gx is a $\forall_{a\to a\to a} g \forall_a x$-term of type $a \to a$. There is exactly one solution for this unification problem, namely,

$$\{(F, \lambda w(gwx)), (Z, g(k(\lambda w(gwx)))x)\}.$$

If this solution is deskolemized, we get a solution for the original unification problem, namely, $\{(F, \lambda w(gwx)), (Z, gyx)\}$.

Example 4.14. *If we return to Example 4.3 and make certain that the type of the*

Skolemized variable is nonempty, we notice that deskolemizing will work correctly. Consider the modified unification problem $\forall_b w \forall_a z \forall_{b\to a} g \exists_a X \forall_b y \top$ and its Skolemized form

$$\forall_b w \forall_a z \forall_{b\to a} g \forall^1_{a\to b} f \exists_a X \top.$$

This latter unification problem now has solutions $X \mapsto z$, $x \mapsto (gw)$, $X \mapsto g(fz)$, $X \mapsto g(f(g(fz)))$, etc. *We can now deskolemize any of these solutions by replacing the subterms $fz, f(g(fz))$* etc. *with w. All these solutions collapse down to just the two solutions for the original problem, namely $X \mapsto z$ and $X \mapsto (gw)$.*

The more general case of Skolemization for nonprenex normal formulas of a higher-order logic is studied in (Miller, 1987). In the next section, we present the dual to Skolemization mentioned earlier.

5. Raising

The dual operation of rotating an existential quantifiers to the left over a universal quantifier will be called *raising*. In comparison to Skolemization, the metatheory of this rewriting process is particularly simple. It requires no new declarations, such as ranked quantifiers. Solutions before and after raising can be placed in one-to-one correspondence. The following theorem is the main result of this section.

Theorem 5.1. *Let $\mathcal{D}$ be a conjunction of equations so that*

$$\mathcal{Q}_1 \forall_\tau y \exists_\delta x \mathcal{Q}_2 \mathcal{D} \qquad (*)$$

is a unification problem. This problem has a solution if and only if the unification problem

$$\mathcal{Q}_1 \exists_{\tau\to\delta} h \forall_\tau y \mathcal{Q}_2 [hy/x]\mathcal{D} \qquad (**)$$

has a solution. Furthermore, a solution for either problem can be transformed in a one-to-one fashion to a solution of the other problem.

Proof. Let θ be a solution to $(*)$ and let t be θx. Set $\theta' := \theta - \{(x,t)\}$ and $\theta'' := \theta' \cup \{(h, \lambda yt)\}$. Clearly, θ'' is a $\mathcal{Q}_1\exists h\forall y\mathcal{Q}_2$-substitution. Also, $\theta''([hy/x]\mathcal{D})$ is equal to $[(\lambda yt)y/x](\theta'\mathcal{D})$ which is nothing more than $\theta\mathcal{D}$. Hence, θ'' is a solution for $(**)$.

For the converse, let θ be a solution for $(**)$. Let t be the value of θh, and set $\theta' := \theta - \{(h,t)\}$ and $\theta'' := \theta \cup \{(x, ty)\}$. Clearly, θ'' is a $\mathcal{Q}_1\forall y\exists x\mathcal{Q}_2$-substitution. Now, $\theta[hy/x]\mathcal{D}$ is equal to $[ty/x]\theta'\mathcal{D}$ which is the same as $\theta''\mathcal{D}$. Hence, θ'' is a solution to $(*)$.

The translations of solutions given in this proof put the solutions for $(*)$ in one-to-one correspondence with the solutions for $(**)$. □

Definition 5.2. *If P is the unification problem $(*)$ and P' is the unification problem $(**)$ above, we say that P' is the result of* raising *P at $\exists_\delta x$.*

Example 5.3. *Let a be a primitive type. The unification problem*

$$\forall_{a\to a\to a} f \forall_a y \exists_{a\to a} X \forall_a z [Xz \overset{a}{=} fzy]$$

can be raised twice to

$$\exists_{(a\to a\to a)\to a\to a\to a} H \forall_{a\to a\to a} f \forall_a y \forall_a z [Hfyz \overset{a}{=} fzy].$$

Using Proposition 2.7, this is solvable if and only if

$$\exists_{(a\to a\to a)\to a\to a\to a}H[\lambda f\lambda y\lambda z.Hfyz = \lambda f\lambda y\lambda z.fzy]$$

is provable. Here the type on this last equation is $(a \to a \to a) \to a \to a \to a$*. Since the left-hand side of the equation is* η*-convertible to* H*, the only solution to this unification problem is* $\{(H, \lambda f\lambda y\lambda z.fzy)\}$*. By the proof Theorem 5.1, we know that the original unification problem has exactly one solution, namely* $\{(X, \lambda z.fzy)\}$*.*

Raising is closely related to a translation used in (Statman, 1981) where several simplifications in the presentation of unification problems are made. First, a unification problem with a $\exists\forall$-prefix of the form

$$\exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_p[t_1 = s_1 \wedge \dots \wedge t_m = s_m] \qquad (*)$$

is encoded by the two λ-terms

$M_1 := \lambda x_1 \dots \lambda x_n \lambda y_1 \dots \lambda y_p \lambda z(zt_1 \dots t_m)$ and
$M_2 := \lambda x_1 \dots \lambda x_n \lambda y_1 \dots \lambda y_p \lambda z(zs_1 \dots s_m)$.

The unification problem in $(*)$ has a solution if and only if there is a list of λ-terms, $N_1, \dots, N_n$ such that for all $i = 1, \dots, n$, N_i can be given the same type as x_i is given in the prefix and such that

$$M_1N_1 \dots N_n \ \beta\eta\text{-converts to } M_2N_1 \dots N_n. \qquad (**)$$

A second simplification in (Statman, 1981) is that constants are not allowed in the terms M_1 and M_2. Assume that the constants $c_1, \dots, c_q$ $(q \geq 0)$ occurred in $(*)$ and hence in the terms M_1 and M_2. Let $z_1, \dots, z_q$ be variables that have no occurrences in $(*)$ and let φ be the mapping that when applied to a λ-term, replaces the constant c_i with z_i for all $i = 1, \dots, q$. Instead of using M_1 and M_2 to denote the unification problem $(*)$, Statman uses the following two constant-free terms:

$M_1^* := \lambda w_1 \dots \lambda w_n \lambda z_1 \dots \lambda z_q((\varphi M_1)(w_1z_1 \dots z_q) \dots (w_nz_1 \dots z_q))$ and
$M_2^* := \lambda w_1 \dots \lambda w_n \lambda z_1 \dots \lambda z_q((\varphi M_2)(w_1z_1 \dots z_q) \dots (w_nz_1 \dots z_q))$.

It is easy to show $(**)$ is satisfied if and only if

$$M_1^*N_1^* \dots N_n^* \ \beta\eta\text{-converts to } M_2^*N_1^* \dots N_n^*,$$

where, for $i = 1, \dots, n$, N_i^* is equal to $\lambda z_1 \dots \lambda z_q \varphi N_i$. In this sense, it is always possible to replace constants appearing in unification problems with abstracted variables.

The relationship of this translation of unification problems to raising is immediate. If the unification problem $(*)$ contained the constants $c_1, \dots, c_q$, we would have chosen to declare them explicitly in the prefix of $(*)$; that is, we would have written the unification problem as

$$\forall z_1 \dots \forall z_q \exists x_1 \dots \exists x_n \forall y_1 \dots \forall y_p[\varphi t_1 = \varphi s_1 \wedge \dots \wedge \varphi t_m = \varphi s_m].$$

By repeatedly raising its existential variables, this problem could be written into the form

$$\exists w_1 \dots \exists w_n \forall z_1 \dots \forall z_q \forall y_1 \dots \forall y_p[\rho t_1 = \rho s_1 \wedge \dots \wedge \rho t_m = \rho s_m],$$

where, for $i = 1, \dots, n$, w_i is the result of raising x_i by moving it left over $\forall z_1 \dots \forall z_q$, and ρ is the substitution that results in applying φ and then replacing x_i with $(w_iz_1 \dots z_q)$

for $i = 1, \ldots, n$. This latter unification problem is represented by the pair of λ-terms M_1^* and M_2^*.

The operation of $\forall$-*lifting* in (Paulson, 1989) is also related to raising in the sense that it can be presented as the backchaining inference rule followed by a sequence of raising steps (see (Miller, 1991b)).

6. Simplifying the Prefix of a Unification Problem

Skolemization can be used to rewrite a unification problem into a problem with a $\forall\exists$-prefix: repeatedly move universal quantifiers left over existential quantifiers increasing their rank and their type as they move. Once a $\forall\exists$-prefix is constructed, all the universal quantifiers can be replaced by "ranked" constants. The resulting unification problem is purely existential and as such can be dealt with by conventional unification processes, modified if necessary to handle such ranked constants. Since λ-abstractions are not available in the first-order setting, first-order unification does not need to be so modified. We show in the next section how the unification of λ-terms must be modified to deal correctly with ranked constants. Once a solution for the simplified unification problem is found, it can be converted to a solution of the original problem by repeatedly deskolemizing the solution, provided that all the types of Skolemized quantifiers are not empty.

In a similar manner, any prefix without positive ranked universal quantifiers can be rewritten using raising until it is an $\exists\forall$-prefix. The resulting unification problem can further be simplified by taking the universally quantified variables in the prefix and λ-abstracting them over the terms in all the equations of the problem. The resulting unification problem again has only an existential prefix, with terms containing possibly long binders and high orders. For each universal quantifier an existential quantifier is moved over, the type of the variable being quantified is raised by giving it an additional argument. Thus, a unification problem that is first-order, that is, in which the existentially quantified variables are all of primitive type, would be converted to a unification problem with functional variables. This may seem undesirable given that several useful properties of first-order unification do not hold for unification involving functional variables. As shown in Section 12, the complexity introduced by raising is particularly simple. A raised first-order unification problem has a decidable unification problem and most general unifiers.

The two rewriting processes, Skolemization and raising, can be used together in the same prefix. Once a universal quantifier is moved left by Skolemization, no existential variable can be move left past it since raising can only involve universal quantifiers of rank zero. Prefixes of the form $\forall\exists\forall$ represents a natural reduction class for unification problems: the outermost universal quantified variables denote the constants of the problem while the inner most universal quantified variables denote the top-level λ-abstractions.

Of course, it is should be possible to take existing unification algorithms and modify them so that they can deal directly with general unification problems. This is the topic of the next section.

7. Reducing Unification Problems

This section is an adaptation of part of (Huet, 1975) to our mixed prefix setting. Several rewriting methods are presented that convert unification problems to, hopefully, more solvable problems.

Following standard convention, equations occurring within unification problems will also be called *disagreement pairs*. The following classification of prefixed terms and disagreement pairs is central to the unification process describe below.

Definition 7.1. *Any β-normal prefixed term $\mathcal{Q}t$ can be written uniquely as*

$$\mathcal{Q}\lambda x_1 \ldots \lambda x_n(yt_1 \ldots t_m)$$

where $n \geq 0$, $m \geq 0$, and y is a variable. The binder *of this term is the list $x_1, \ldots, x_n$, its* head *is the variable y, and its* arguments *are the terms $t_1, \ldots, t_m$.*

If y is existentially quantified in $\mathcal{Q}$, then $\mathcal{Q}t$ is flexible. *Otherwise, y is either universally bound in $\mathcal{Q}$ or a member of the binder of t: in either case $\mathcal{Q}t$ is* rigid.

Let P be a unification problem with prefix $\mathcal{Q}$ and let $t \overset{\tau}{=} s$ be an equation of P. (Of course, τ is a primitive type.) This equation is rigid-rigid *if both $\mathcal{Q}t$ and $\mathcal{Q}s$ are rigid, is* flexible-flexible *if both $\mathcal{Q}t$ and $\mathcal{Q}s$ are flexible, and is* flexible-rigid *otherwise.*

This classification is important because it captures the one invariance about substitution and β-normalization that we will use repeatedly: if $\mathcal{Q}t$ is rigid, then any $\mathcal{Q}$-substitution instance of $\mathcal{Q}t$ is rigid and has the same head as $\mathcal{Q}t$. This invariance justifies the following simplification of rigid-rigid disagreement pairs. Let $ht_1 \ldots t_p = ks_1 \ldots s_q$ be a rigid-rigid disagreement pair in a unification problem with prefix $\mathcal{Q}$. If these two terms are to have instances that are λ-convertible under some substitution, their heads must be the same, p and q must be equal, and for $i = 1, \ldots, p$, t_i and s_i must be simultaneously unifiable under the prefix $\mathcal{Q}$.

Example 7.2. *Using these observations, we can rewrite the unification problem*

$$\forall_{((i\rightarrow i)\rightarrow i)\rightarrow i\rightarrow i} f \forall_i a \exists_i V \exists_{i\rightarrow i} W \forall_i b[f(\lambda x(xa))V \overset{i}{=} f(\lambda y(yV))(Wb)]$$

into the term

$$\forall f \forall a \exists V \exists W \forall b[\lambda x(xa) \overset{(i\rightarrow i)\rightarrow i}{=} \lambda y(yV) \wedge V \overset{i}{=} Wb].$$

This is not a proper unification problem. Using Proposition 2.7, this term has the same solutions as the unification problem

$$\forall f \forall a \exists V \exists W \forall b \forall_{i\rightarrow i} x[xa \overset{i}{=} xV \wedge V \overset{i}{=} Wb].$$

This problem contains a rigid-rigid pair and can be rewritten as

$$\forall f \forall a \exists V \exists W \forall b \forall x[a \overset{i}{=} V \wedge V \overset{i}{=} Wb].$$

This final unification problem contains no rigid-rigid disagreement pairs and cannot be simplified in this manner any further. The first pair is flexible-rigid while the second is flexible-flexible.

Definition 7.3. *Let P be a unification problem that contains an equation of the form $kt_1 \ldots t_n = ks_1 \ldots s_n$ $(n \geq 0)$. The process of rewriting P by replacing this equation with the conjunction $t_1 = s_1 \wedge \ldots \wedge t_n = s_n$ and then using Proposition 2.7 repeatedly to make the resulting term a unification problem is called the* simplification *rewriting step. If $n = 0$ then $\top$ replaces the equation $k = k$. A unification problem that contains no rigid-rigid pairs with the same head is a* simplified *unification problem.*

The following proposition follows immediately.

PROPOSITION 7.4. *Using simplification and the rewriting rules (1), (2), (4), and (5) of Proposition 2.7, a unification problem P can be rewritten to a unification problem, say P', in which there are either no rigid-rigid disagreement pairs or where some rigid-rigid disagreement pair is composed of terms with different heads. In either case, the solutions to P' are exactly the same as the solutions to P. Furthermore, P' is unique modulo α-conversion and the ordering of the rightmost universal variables of the prefix.*

Further analysis of disagreement pairs will require the use of substitutions for existentially quantified variables. These substitutions will not, in general, be $\mathcal{Q}$-substitutions (where $\mathcal{Q}$ is the prefix of the unification problem in question) since such substitutions replace existential variables with essentially *closed* or completed terms. We shall need the flexibility in building solutions incrementally by using substitution terms that are *open*, that is, that contain subterms that must be determined later. To this end, we introduce the following definitions.

DEFINITION 7.5. *Let $\mathcal{Q}t$ be a prefixed term where $\mathcal{Q}$ is of the form $\mathcal{Q}_1\exists_\tau f\mathcal{Q}_2$. A prefixed term $\mathcal{Q}_1\exists H_1\ldots\exists H_n s$ ($n \geq 0$) is* $\mathcal{Q}$-free for f *if it is of type τ. Here, we may assume that the variables $H_1,\ldots,H_n$ are not bound in $\mathcal{Q}$. The* result of substituting s for f in $\mathcal{Q}t$, *written as $[f \mapsto s]\mathcal{Q}t$, is the prefixed term*

$$\mathcal{Q}_1\exists H_1\ldots\exists H_n\mathcal{Q}_2([s/f]t).$$

Similarly, if P is a unification problem, it can also be thought of as a prefixed term. Thus, the term $[f \mapsto s]P$ can be made into a unification problem by using β-reduction and the rewrites in Proposition 2.7.

If $\mathcal{Q}$ is of the form $\mathcal{Q}_1\exists_\tau f\mathcal{Q}_2$ and if $\mathcal{Q}_1\exists H_1\ldots\exists H_n s$ is $\mathcal{Q}$-free for f, the substitution $[f \mapsto s]$ is not generally a $\mathcal{Q}$-substitution because s may contain existential variables of $\mathcal{Q}_1$ or it may contain new existential variables (namely, $H_1,\ldots,H_n$). We shall make use of these kinds of substitutions, however, to build solutions for unification problems. The substitutions of the form $[f \mapsto s]$ used in this section will all be such that free variables of s are either universal variables of $\mathcal{Q}_1$ or are new existential variables. It will not be until Section 10 that such substitution terms will contain existential variables of $\mathcal{Q}_1$.

DEFINITION 7.6. *Let σ be a $\mathcal{Q}_1\exists H_1\ldots\exists H_n\mathcal{Q}_2$-substitution and let $\mathcal{Q}_1\exists H_1\ldots\exists H_n s$ be a prefixed term that is $\mathcal{Q}_1\exists_\tau f\mathcal{Q}_2$-free for f. The composition of $[f \mapsto s]$ with σ, written $[f \mapsto s]\circ\sigma$, is the $\mathcal{Q}_1\exists f\mathcal{Q}_2$-substitution given by*

$$\{(f, s')\} \cup \{(v, \sigma v) \mid v \in dom(\sigma) \text{ and } v \notin \{H_1,\ldots,H_n\}\},$$

where s' is the term part of the prefixed term $\sigma\mathcal{Q}_1\exists H_1\ldots\exists H_n s$. Composition associates to the right and satisfies the equation $(f\circ g)x = g(f(x))$.

We shall occasionally suppress stating the prefix of $\mathcal{Q}_1\exists H_1\ldots\exists H_n s$ since it can be constructed from $\mathcal{Q}$ by taking the free variables of s not bound to the left of f as the variables $H_1,\ldots,H_n$ and then determining an appropriate type for those variables. The following proposition shows that when s is $\mathcal{Q}$-free for f, $[f \mapsto s]$ can represent a component of a final complete solution.

Proposition 7.7. *Let P be a unification problem with prefix $\mathcal{Q} = \mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2$. Let s be $\mathcal{Q}$-free for f. Then $[f \mapsto s]P$ has solution σ if and only if P has solution $[f \mapsto s] \circ \sigma$.*

Proof. Let $\mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2 t$ be a prefixed term and let $\mathcal{Q}_1 \exists H_1 \ldots \exists H_n s$ be $\mathcal{Q}$-free for f. Let σ be a $\mathcal{Q}_1 \exists H_1 \ldots \exists H_n \mathcal{Q}_2$-substitution. The β-normal form of the formulas $\sigma([f \mapsto s]\mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2 t)$ and $([f \mapsto s] \circ \sigma)\mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2 t$ are the same. Let P be a unification problem with prefix $\mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2$. Since $\sigma([f \mapsto s]\mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2 t)$ and $([f \mapsto s] \circ \sigma)\mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2 t$ are the same, $[f \mapsto s]P$ has solution σ if and only if P has solution $[f \mapsto s] \circ \sigma$. □

We can now return to the construction of solutions to unification problems. Consider the case where only one term of a disagreement pair is rigid; the other one being flexible. For example, let $ft_1 \ldots t_p = ks_1 \ldots s_q$ be a disagreement pair and assume that f is existentially quantified while k is a universally quantified. For these to be unifiable, f must be instantiated with a term that, after β-normalization, puts k at the head of the flexible term. There are two general ways to place k at the head since there are two processes that are used to make these terms equal, namely, substitution *and* λ-reduction.

Straightforward substitution can make the flexible term look like the rigid one if the substitution term for f has head k. That is, the substitution term *imitates* the head of the rigid term. However, λ-reduction can also be performed on disagreement pairs after a substitution. Thus the term substituted for f could be used to migrate subterms of the flexible term that might contain k into the head position of the flexible term. This method of placing k at the head of a the flexible term is indirect and, hence, will often produce several possible substitution terms. Immediate subterms in the flexible term can be accessed by a sequence of *projections*. For example, to place the i^{th} argument t_i of the flexible term in the head position, a substitution term for f would be an abstraction of the variables $\lambda w_1 \ldots \lambda w_p$ over a term with w_i as its head. The result of substituting such a term for f and then normalizing will leave a term whose head is equal to the head of t_i. If the head of t_i is k then we have succeeded in our goal. If the head is a universally quantified variable different than k, we have failed and need to backtrack to other choices. If the head of t_i is an existentially quantified variable, we could use such a projection term again to access subterms of t_i, or try to use imitation on the flexible head of t_i.

Let us now be more precise about both the imitation and projection schemes for getting a flexible head to match a rigid head. Assume that we are given the flexible-rigid disagreement pair $ft_1 \ldots t_p = ks_1 \ldots s_q$ where f is existentially quantified and k is universally quantified.

If k is quantified to the right of f, the imitation scheme cannot be used since the substitution term for f cannot contain k. If k is quantified to the left of where f is quantified, then it is possible to make the head of the flexible term become k directly. Let r be the rank of k. If $r = 0$ then the imitation term for f is a λ-term of the form

$$\lambda w_1 \ldots \lambda w_p \, (k(H_1 w_1 \ldots w_p) \ldots (H_q w_1 \ldots w_p)),$$

which has the same type as f. This term takes each of the p arguments of the flexible term and constructs a new term with k as the head and whose arguments depend functionally on the arguments of the flexible term. Notice that this functional dependency is left unspecified; $H_1, \ldots, H_q$ are unspecified higher-order variables. Examining just the surface structure of the rigid term does not provide any information on how these new variables should be instantiated. Their determination is attempted later after simplification removes more of the surface structure.

Notice that if the rank r is positive then the substitution term suggested above is not a valid $\mathcal{Q}$-term. This is easily corrected, however, by restricting the first r arguments of this substitution term to not be functionally dependent on the abstracted variables. In particular, the correct substitution term would be

$$\lambda w_1 \ldots \lambda w_p \; (kH_1 \ldots H_r (H_{r+1} w_1 \ldots w_p) \ldots (H_q w_1 \ldots w_p)).$$

Since we have assumed that the terms in unification problems are $\mathcal{Q}$-terms themselves, $r \leq q$ and this term is sensible.

In either case, the displayed term is called the *imitation term* for the given disagreement pair. We shall require that the free variables $H_1, \ldots, H_q$ do not appear in the given unification problem's prefix. Notice that an imitation term is unique up to the choice of these free variables and the bound variables $w_1, \ldots, w_p$.

Finally, we need to consider the projection scheme for matching a flexible head to a rigid head. In this case, we only try to rearrange the flexible term so that one of its arguments is moved into the head position. In particular, consider all the terms of the same type as f that are of the form

$$\lambda w_1 \ldots \lambda w_p \; (w_i (H_1 w_1 \ldots w_p) \ldots (H_m w_1 \ldots w_p)),$$

where $m \geq 0$, $1 \leq i \leq p$, and $H_1, \ldots, H_m$ are variables that do not occur in the prefix of the given unification problem. All such terms are $\mathcal{Q}$-terms and are never of primitive type. There are, of course, p or fewer such terms, if we do not count differences in the names of their free and bound variables. Such a term instructs the flexible term to project its i^{th} argument to its head and to give it any additional arguments (the m arguments in the term above) it might need for the entire term to be the same type of function as f. A *projection term* for the given flexible-rigid disagreement pair is any of these terms.

Notice that all of the substitution terms above are designed to be functionally dependent on all p arguments of the flexible term. Such terms need not be functional on all such arguments, but if η-conversion is available, then we only need to consider the terms built here. If η-conversion is not available, as in the first part of (Huet, 1975), then more imitation and projection terms need to be considered.

Definition 7.8. *Let P be a unification problem with prefix $\mathcal{Q} = \mathcal{Q}_1 \exists_\tau f \mathcal{Q}_2$ and with a flexible-rigid disagreement pair $ft_1 \ldots t_p = ks_1 \ldots s_q$. We shall write imit$(f, \tau, k, \mathcal{Q})$ to be either the empty set if k is bound to the right of f or the set $\{\mathcal{Q}_1 \exists H_1 \ldots \exists H_q t\}$ where t is the imitation term described above. Similarly, we write proj$(f, \tau, \mathcal{Q})$ to be the set of all projection terms generated above for type τ with the appropriate prefix added. Finally, we define match$(f, \tau, k, \mathcal{Q}) =$ imit$(f, \tau, k, \mathcal{Q}) \cup$ proj$(f, \tau, \mathcal{Q})$. Any term in this set is a* match term *for the disagreement pair $ft_1 \ldots t_p = ks_1 \ldots s_q$. The values of imit, proj, and match are unique up to alphabetic changes of bound variables. The argument $\mathcal{Q}$ is used during building the imitation term to determine the rank of k and whether or not k is to the right or left of f. It is also used to constrain the picking of the variables $H_1, \ldots, H_n$ since these cannot appear in $\mathcal{Q}$.*

Example 7.9. *Let i and a be primitive types. Let $\mathcal{Q}$ be $\forall k \forall^1 l \exists F \forall j$ where the variables k, j all have type $i \to i$ and l has the type $i \to i \to i \to i$. The following are some values of imit. (The prefixes for the various match terms presented below are suppressed: they are* *all of the form $\forall k \forall^1 l \exists H_1 \ldots \exists H_i$ where $i \geq 0$ is the number of new existential variables free in the term.)*

$$\begin{aligned} imit(F, i, k, \mathcal{Q}) &= \{kH_1\} \\ imit(F, i \to i, k, \mathcal{Q}) &= \{\lambda w(k(H_1 w))\} \\ imit(F, i \to a, k, \mathcal{Q}) &= \{\} \\ imit(F, i \to i, j, \mathcal{Q}) &= \{\} \\ imit(F, (i \to i) \to i \to i, l, \mathcal{Q}) &= \{\lambda w_1 \lambda w_2 (l H_1 (H_2 w_1 w_2)(H_3 w_1 w_2))\} \end{aligned}$$

In this last imitation term, the types of w_1 and w_2 are $i \to i$ and i, while the types of H_1, H_2, and H_3 are $i, (i \to i) \to i \to i$ and $(i \to i) \to i \to i$, respectively. Below are some values of proj.

$$\begin{aligned} proj(F, i \to i \to a, \mathcal{Q}) &= \{\} \\ proj(F, i \to i, \mathcal{Q}) &= \{\lambda w\, w\} \\ proj(F, i \to i \to i, \mathcal{Q}) &= \{\lambda w_1 \lambda w_2\, w_1, \lambda w_1 \lambda w_2\, w_2\} \\ proj(F, (i \to i) \to i \to i, \mathcal{Q}) &= \{\lambda w_1 \lambda w_2\, (w_1(H_1 w_1 w_2)), \lambda w_1 \lambda w_2\, w_2\} \end{aligned}$$

In order to facilitate a completeness proof for this unification rewriting process, we define the following complexity measures on terms and substitutions.

Definition 7.10. *Let t be a β-normal term. Let $|t|$ denote the number of occurrences of applications in t. That is,*

$$|\lambda x_1 \ldots \lambda x_n (h t_1 \ldots t_m)| = m + \sum_{i=1}^{m} |t_i| \qquad (n, m \geq 0).$$

Let σ be the substitution $\{(w_1, t_1), \ldots, (w_n, t_n)\}$ where $n \geq 0$. Then $|\sigma|$ is defined as

$$|\sigma| = n + \sum_{i=1}^{n} |t_i|.$$

The following proposition is critical for establishing a completeness theorem later.

Proposition 7.11. *Let P be a simplified unification problem that has solution σ. Also assume that the prefix $\mathcal{Q}$ of P is of the form $\mathcal{Q}_1 \exists_\tau F \mathcal{Q}_2$ and that P contains a flexible-rigid disagreement pair with F as the flexible head and k as the rigid head. Then there exists a unique prefixed term $\mathcal{Q}_1 \exists H_1 \ldots \exists H_n s$ in $match(F, \tau, k, \mathcal{Q})$ $(n \geq 0)$ and a unique $\mathcal{Q}_1 \exists H_1 \ldots \exists H_n$-substitution ρ with domain $\{H_1, \ldots, H_n\}$ such that*

(1) *σF β-converts to ρs.*

Furthermore, let σ' be the substitution $\rho \cup (\sigma - \{(F, \sigma F)\})$. Then

(2) *$\sigma = [F \mapsto s] \circ \sigma'$,*
(3) *σ' is a solution to the unification problem $[F \mapsto s]P$, and*
(4) *$|\sigma'| < |\sigma|$.*

Proof. Let t be the β-normal form of σF. Then t is of the form $\lambda w_1 \ldots \lambda w_n (c t_1 \ldots t_m)$ for some variables $c, w_1, \ldots, w_n$ and terms $t_1, \ldots, t_m$ $(n, m \geq 0)$. Since t is $\mathcal{Q}$-closed for F, the variable c must be either universally bound in $\mathcal{Q}$ or a member of the list $w_1, \ldots, w_m$. In the first case, the head of the flexible term will become c. Since the unification problem P has a solution, c must be k. Let s be the imitation term for this flexible-rigid pair. If c is a member of the list $w_1, \ldots, w_m$, then by the construction of match terms, there is a unique term, say $s \in match(F, \tau, k, \mathcal{Q})$, which projects the same member of the binder to its head. In either case, let $H_1, \ldots, H_p$ be the new free variables of s, listed in the left-to-right order they appear in s $(p \geq 0)$. If we define $\rho = \{(H_i, \lambda w_1 \ldots \lambda w_m t_i) \mid i = 1, \ldots p\}$, we see that t is β-convertible to ρs (hence, showing (1) above). Notice also that $|t| = m + \sum_{i=1}^{m} |t_i|$. Let σ' be as defined in the theorem. Then (2) obviously holds. Since $([f \mapsto s] \circ \sigma')P$ is equal to the prefixed term σP, σ' is a solution to $[x \mapsto s]P$ (showing (3)). Finally,

$$|\sigma'| = |\sigma| - |t| + \sum_{i=1}^{m} |t_i| - 1 + m$$

which simplifies to $|\sigma'| = |\sigma| - 1$ (conclusion (4) above). □

Definition 7.12. *A unification problem containing either no equations or only flexible-flexible equations is called a* flexible-flexible *unification problem.*

In Section 8, we consider the general problem of determining when flexible-flexible problems have solutions. Consider the following example of a flexible-flexible unification problem that does not have a solution.

Example 7.13. *Let a, b, and c be primitive types. Consider the following flexible-flexible unification problem:*

$$\forall_{a\to b} y \exists_{a\to b} F \forall_{c\to b} z \exists_{c\to b} G \forall_a u \forall_c v \; [Fu \overset{b}{=} Gv].$$

Although all existentials in this prefix are nonempty, this unification problem has no solution.

We can now organize the two rewriting rules of simplification and substitution with matching terms into a search process for flexible-flexible unification problems. Such a search is often referred to as *pre-unification.*

Definition 7.14. *Let P and P' be unification problems. We shall write $P \xrightarrow[s]{F} P'$ if all the following conditions are true.*

1. *P is in simplified form,*
2. *F is an existentially quantified in P,*
3. *there is a flexible-rigid pair in P in which the flexible term has F as its head,*
4. *s is a matching term for this pair, and finally,*
5. *P' is the simplified form of $[F \mapsto s]P$.*

These rewriting rules can be used to nondeterministically decompose unification problems with solutions to flexible-flexible unification problems with solutions.

PROPOSITION 7.15. *Let P be a simplified unification problem. P has a solution σ if and only if there exists a sequence of rewrite steps*

$$P \underset{s_1}{\overset{F_1}{\rightarrow}} \dots \underset{s_n}{\overset{F_n}{\rightarrow}} P' \qquad (n \geq 0)$$

where P' is a flexible-flexible problem with solution ρ and $\sigma = [F_1 \mapsto s_1] \circ \cdots \circ [F_n \mapsto s_n] \circ \rho$.

PROOF. Let P be a simplified unification problem. If a sequence of unification problems as above carries P to a flexible-flexible problem with solution ρ, then by induction and Propositions 7.4 and 7.7, P has the solution σ, defined in the theorem. Assume the converse, however, that P has a solution, say σ. Then a sequence of such rewriting steps must exist and its length is bounded by $|\sigma|$. The following simple nondeterministic procedure will construct such a sequence. Let σ_1 be σ, let P_1 be P, and pick any flexible-rigid pair in P. Let F_1 be the head of the flexible term. By Proposition 7.11, this pair determines a nonempty set of matching terms that contains a unique term s_1 such that $[F_1 \mapsto s_1]P$ has a solution σ_2 where $|\sigma_2| < |\sigma|$ and $\sigma = [F_1 \mapsto s_1] \circ \sigma_2$. Let P_2 be a simplified form of $[F_1 \mapsto s_1]P$. If P_2 is a flexible-flexible problem, then we are finished. Otherwise, we repeat this selection of a flexible-rigid pair and a match term again on this new unification problem. This process must terminate since each subsequent call to the unification process will reduce the complexity of a solution the unification problem. □

N.B. In the rest of this paper, we shall assume that unification problems will not contain universal quantifiers of positive rank. This assumption is taken to simplify the presentation of various different technical definitions in the following sections. While most of these definitions can be generalized to handle positively ranked universal quantifiers, doing so complicates those definitions without adding to their content.

8. Solving Flexible-Flexible Problems

Flexible-flexible unification problems are considered "reduced" in the sense that the principal invariant used in Section 7, that rigid terms preserve their heads under substitution, is not useful when examining flexible-flexible unification problems. We now show that deciding whether or not a flexible-flexible unification problem has a solution is undecidable. For the next three lemmas and one theorem, let i be a primitive type. Proofs for the lemmas are straightforward and not given.

LEMMA 8.1. *Let t be a closed β-normal term of type $(i \to i) \to i \to i$. Then t is η-convertible to a term of the form $\lambda f \lambda x. f^n x$ for some $n \geq 0$.*

Closed λ-terms of type $(i \to i) \to i \to i$ are called *Church numerals.* For $n \geq 0$, we write $\hat{n}$ to denote the n^{th} Church numeral, that is, the term $\lambda f \lambda x. f^n x$. For example, $\hat{0} = \lambda f \lambda x. x$, $\hat{1} = \lambda f \lambda x. f x$, and $\hat{2} = \lambda f \lambda x. f(f x)$.

LEMMA 8.2. *The flexible-flexible unification problem*

$$\exists_{(i \to i) \to i \to i} N \forall_{i \to i} f \forall_i x [N f (N f x) \overset{i}{=} (N f (f x))]$$

has the unique solution $\{(N, \hat{1})\}$.

Lemma 8.3. *Let the existential quantifiers below associate type $(i \to i) \to i \to i$ to their bound variables. The flexible-flexible unification problem*

$$\exists M \exists N \exists P \forall_{i \to i} f \forall_i x[(Nf)(Mf)x \overset{i}{=} Pfx]$$

has solution $\{(N, r), (M, s), (P, t)\}$ if and only if for some $n, m \geq 0$, r is $\hat{n}$, s is $\hat{m}$, and t is $\widehat{n+m}$. The flexible-flexible unification problem

$$\exists M \exists N \exists P \forall_{i \to i} f \forall_i x[N(Mf)x \overset{i}{=} Pfx]$$

has solution $\{(N, r), (M, s), (P, t)\}$ if and only if for some $n, m \geq 0$, r is $\hat{n}$, s is $\hat{m}$, and t is $\widehat{n \times m}$.

The outline of the following proof is suggested by the main proof in (Goldfarb, 1981).

Theorem 8.4. *The problem of determining if a given flexible-flexible unification problem has a solution is recursively undecidable.*

Proof. Using the preceding three lemmas, it is simple to encode any finite set of equation of the form $x = 1, x+y = z$, and $x \times y = z$ into a flexible-flexible unification problem such that the unification problem has a solution if and only if the set of equations has an interpretation of its variables over nonnegative integers such that all the equations are true. In this encoding, each variable in an equation would correspond to an existential variable of type $(i \to i) \to i \to i$, and each equation would correspond to a flexible-flexible disagreement pair. Since the problem of determining if such a set of equations can be solved over the nonnegative integers is recursively unsolvable (Hilbert's Tenth Problem), the problem of determining if flexible-flexible unification problems have solutions is recursively unsolvable. Notice that the unification problems needed for this encoding are only second-order; that is, the highest order of any variables in these prefixes is 2. □

There are many flexible-flexible unification problems for which it is easy to compute a solution. The following proposition describes such a collection of flexible-flexible unification problems.

Proposition 8.5. *Let P be a flexible-flexible unification problem with prefix $\mathcal{Q}$. Let $\mathcal{E}$ be the set of existential variables of $\mathcal{Q}$, and let $\approx^0$ be the relation on $\mathcal{E}$ such that $x \approx^0 y$ if and only if x and y are the heads of a common disagreement pair of P. Let $\approx$ be the equivalence closure of $\approx^0$. Let E be some $\approx$-equivalence class, let τ be the common target type of the variables in E, and let $\mathcal{Q}'$ be the prefix to the left of the leftmost variable in E. Call the set E* solvable *if there is a $\mathcal{Q}'$-term of type τ. Finally, if all equivalence classes of $\mathcal{E}$ are solvable, then P has a solution.*

Proof. The following proof is a simple extension of a proof given in (Huet, 1975). Let E be an equivalence class of existential variables of the prefix $\mathcal{Q}$, let τ be their common target type, let $\mathcal{Q}'$ be the prefix to the left of the leftmost variable in E, and let t be a $\mathcal{Q}'$-term of type τ. For each variable x in E substitute the λ-term $\lambda w_1 \ldots \lambda w_n.t$, where n is the unique nonnegative integer needed to make this term the same type as x and the variables $w_1, \ldots, w_n$ are not free in t. In this way, a substitution for all the existential variables of $\mathcal{Q}$ can be built. It is immediate that this substitution is a solution for P. □

In the event that a unification problem has a $\forall\exists$-prefix, then the above proposition reduces to simply checking if each of the primitive types labeling equations in the unification problem are nonempty with respect to the purely universal part of the prefix.

When flexible-flexible unification problems have solutions, computing all solution can be difficult. Although Proposition 8.5 can be used occasionally to produce some solutions to some flexible-flexible unification problems, it provides no information on the nature of other solutions available. Such unification problems may have an infinite number of incomparable solutions and no nonredundant enumeration of a complete set of solutions for general flexible-flexible unification problem is possible (Huet, 1975).

9. Some Unsolvable Unification Problems

We now identify some classes of unification problems that have no solutions.

Definition 9.1. *A* simple failure problem *is a simplified unification problem containing a disagreement pair that is either* (*i*) *rigid-rigid (thus, its terms have different heads), or* (*ii*) *flexible-rigid and the set of matching terms for this pair is empty.*

Theorem 9.2. *If P is a simple failure node then it has no solutions.*

Proof. Let P be a simple failure unification problem. There are two cases to consider. If P contains a rigid-rigid disagreement pair with different heads, then no substitution into these terms can make them equal and P can have no solution. From Proposition 7.11 it follows that if P has a solution, every flexible-rigid disagreement pair in P must have a nonempty set of *match* terms. Thus, if P contains a flexible-rigid pair that has an empty set of *match* terms, P could not have a solution. □

The analysis required to recognize simple failure unification problems is inexpensive: only the surface structure of the terms involved need to be examined. The remaining failure cases presented in this section are more costly to determine. The following two definitions generally require the examination of much of a term's structure.

Definition 9.3. *Let $\mathcal{Q}t$ be a prefixed formula. The binary relation $\twoheadrightarrow^0$ on prefixed terms is defined by the following two rules. First, $\mathcal{Q}\lambda xt \twoheadrightarrow^0 \mathcal{Q}\forall xt$, provided x is not a member of the prefix $\mathcal{Q}$ (otherwise change the name of x before moving it to the prefix). Second, $\mathcal{Q}(ht_1 \ldots t_n) \twoheadrightarrow^0 \mathcal{Q}t_i$ provided h is a universally quantified variable in $\mathcal{Q}$ and $1 \leq i \leq n$. Let $\twoheadrightarrow$ be the transitive closure of $\twoheadrightarrow^0$. A variable w of $\mathcal{Q}$ is said to have a* permanent occurrence *in $\mathcal{Q}t$ if $\mathcal{Q}t \twoheadrightarrow \mathcal{Q}'(wt_1 \ldots t_n)$ for some prefix $\mathcal{Q}'$, $n \geq 0$, and some terms $t_1, \ldots, t_n$.*

The proof of the following proposition is done by a simple induction on the structure of terms: the proof is omitted.

Proposition 9.4. *Let t be a β-normal $\mathcal{Q}$-term and let y be a universally quantified variable of $\mathcal{Q}$ that has a permanent occurrence in $\mathcal{Q}t$. If σ is a $\mathcal{Q}$-substitution, then y has a permanent occurrence in $\sigma\mathcal{Q}t$. If s is $\mathcal{Q}$-free for f in $\mathcal{Q}$, then y has a permanent occurrence in $[f \mapsto s]\mathcal{Q}t$.*

Definition 9.5. *Let t be a β-normal $\mathcal{Q}$-term. A universally bound variable y of $\mathcal{Q}$* possibly occurs in $\mathcal{Q}t$ *if y occurs free in t or y is to the left of some existential variable of $\mathcal{Q}$ that occurs free in t.*

The following proposition justifies this use of terminology.

Proposition 9.6. *Let P be a unification problem with prefix $\mathcal{Q}$ and a disagreement pair $t = u$. If there exists a universally quantified variable y in $\mathcal{Q}$ that does not have a possible occurrence in t but does have a permanent occurrence in u, then P has no solution.*

Proof. Assume that σ is a solution for P. By Proposition 9.4, y has an occurrence in σu. Since y does not occur in t and since the existential variables free in t are on the left of y, σt does not contain an occurrence of y. Hence, σt and σu are not equal and this contradicts the assumption that σ is a solution to P. □

Example 9.7. *The unification problem*

$$\forall_{i\to i} y \exists_{(i\to i)\to i} H \forall_{i\to i} x [x(Hx) \overset{i}{\doteq} Hy]$$

has no solution since x has a permanent occurrence in $x(Hx)$ while it has no possible occurrence in Hy.

Below we present a different class of unsolvable unification problems. They can be recognized as such by noticing how a simple strategy of applying matching terms would produce a cyclic pattern in the search for a solution.

Definition 9.8. *Let prefix $\mathcal{Q}$ be of the form $\mathcal{Q}_1 \exists x \mathcal{Q}_2$. Consider an equation of the form $xs_1 \ldots s_n = t$ where $n \geq 0$ and for every $i = 1, \ldots, n$, $\mathcal{Q}s_i$ has a head that is bound universally in $\mathcal{Q}_2$. Let Y be the set of heads of the terms $s_1, \ldots, s_n$. Finally, assume that x has a permanent occurrence in t witnessed by the sequence of prefixed terms*

$$\mathcal{Q}t = \mathcal{Q}_1 t_1 \twoheadrightarrow^0 \ldots \twoheadrightarrow^0 \mathcal{Q}_m t_m = \mathcal{Q}'(xe_1 \ldots e_n) \quad (m \geq 2),$$

where for all $i = 1, \ldots, m-1$, the head of t_i is not in the set Y. Such an equation is a simple divergent *equation.*

The following is a generalization of the occurrence-check found with in first-order unification.

Proposition 9.9. *If P is a unification problem with a simple divergent disagreement pair, P has no solution.*

Proof. We shall show that if P is a unification problem with a simple divergent disagreement pair and with a solution, say σ, then there is another unification problem P' with a simple divergent disagreement pair and a solution σ' where $|\sigma'| < |\sigma|$. From this, a contradiction quickly follows: if $n = |\sigma|$, use the above argument $n+1$ times to reach the conclusion that there is some unification problem whose solution has negative size.

Thus, let P contain a simple divergent disagreement pair $xs_1 \ldots s_n = t$, and let Y be as defined in Definition 9.8. Let t be $kt_1 \ldots t_p$ where $p \geq 1$ and let $j \in \{1, \ldots, n\}$ be such that x has a permanent occurrence in the term t_j satisfying the same condition regarding the set Y. By assumption, $k \notin Y$. Since P has a solution, there is a match term s for this equation such that $P \xrightarrow[s]{x} P'$ where P' has a solution σ' with $|\sigma'| < |\sigma|$ and $\sigma = [x \mapsto s] \circ \sigma'$. If s is a projection term, then P' contains a rigid-rigid disagreement pair whose heads are k and the head of s_i for some $i = 1, \ldots, m$. Such a P' is a simple failure problem since these heads are different: if k is in $\mathcal{Q}_1$ then k cannot equal the head of s_i by assumption on s_i, and if k is in $\mathcal{Q}_2$, then k cannot be equal to the head of s_i since it is a member of Y.

Thus s must be the imitation term and k is bound in $\mathcal{Q}_1$. Let the imitation term, s, be written as

$$\lambda y_1 \ldots \lambda y_n (k(H_1 y_1 \ldots y_n) \ldots (H_p y_1 \ldots y_n)).$$

Thus, P' contains the β-normal form of the equation $H_j s_1 \ldots s_n = [s/x]t_j$. Since t_j contained a permanent occurrence of x and the head of s is a universal variable not in Y, this pair is then a simple divergent disagreement pair. This completes the proof. □

Example 9.10. *Consider the unification problem*

$$\forall_{i \to i} a \exists_{i \to i} X \forall_i u [Xu \overset{i}{=} a(Xu)].$$

Since X occurs permanently in $a(Xu)$ and that occurrence is not under u, this unification problem has no solution. Another way to see this is to notice that the imitation term for the disagreement pair in this unification problem is $\lambda w\ .a(Hw)$ for a new variable H of the same type as X. Substituting this into the unification problem yields the problem $\forall a \exists H \forall u [Hu \overset{i}{=} a(Hu)]$, which is simply an alphabetic variant of the original problem. Since this problem cannot be rewritten into a flexible-flexible problem, Proposition 9.1 implies that this problem has no solution.

On the other hand, consider the following unification problem taken from (Huet, 1975):

$$\forall_i a \exists_{(i \to i) \to i} X \forall_{i \to i} u [Xu \overset{i}{=} (u(X(\lambda v\ v)))].$$

Although X occurs rigidly in $u(X(\lambda v\ v))$, this theorem does not apply to this problem since X occurs under u. This unification problem actually does give rise to a flexible-flexible problem since substituting the projection term $\lambda w(w(Hw))$ for X and simplifying yields $\forall a \exists H \forall u [Hu = H(\lambda v\ v)]$. This has the solution, $\{(H, \lambda z\ a)\}$, and the original problem has the solution $\{(X, \lambda w(wa)\}$.

To see why Proposition 9.9 provides a generalization of the occurrence-check in first-order unification, assume that the equation $x = s$ is first-order, x is different than s, and that x occurs in s. Thus, x has a permanent occurrence in s and the restriction on the permanent occurrence is vacuously true. As the proposition concludes, there is no unifier for any unification problem containing this equation.

10. Factors of Unification Problems

In this section, we will capitalize on the fact that occasionally there are unification problems whose solutions all share certain features. Recognizing such cases and features permits a search strategy to commit to a particular approach to constructing an initial portion of a solution without needing to consider backtracking. The common structure of solutions, when they exist, will be called *factors*.

Definition 10.1. *Let P be a unification problem with prefix $\mathcal{Q}$. Let P_0 be P and $\mathcal{Q}_0$ be $\mathcal{Q}$. A list of quadruples*

$$(f_1, s_1, \mathcal{Q}_1, P_1), \ldots, (f_n, s_n, \mathcal{Q}_n, P_n) \quad (n \geq 1)$$

is called a cascading substitution *for P if for $i = 1, \ldots, n$, the term s_i is $\mathcal{Q}_{i-1}$-free for the variable f_i and P_i is the simplified unification problem $[f_i \mapsto s_i]P_{i-1}$ and $\mathcal{Q}_i$ is the prefix of P_i. We shall denote by $[f_1 \mapsto s_1] \circ \cdots \circ [f_n \mapsto s_n]$ the function that carries $\mathcal{Q}_n$-substitutions σ to the $\mathcal{Q}$-substitution $[f_1 \mapsto s_1] \circ \cdots \circ [f_n \mapsto s_n] \circ \sigma$. We shall frequently refer to just the expression $[f_1 \mapsto s_1] \circ \cdots \circ [f_n \mapsto s_n]$ as a cascading substitution. The syntactic variable ψ will be used to denote cascading substitutions. If ψ' denotes the above cascading substitution and if ψ'' denotes a cascading substitution for P_n, then their composition, written as $\psi' \circ \psi''$, is the concatenation of their list of quadruples. The interpretation of $\psi' \circ \psi''$ as a mapping on substitutions is given by composing the meaning of the interpretation of both ψ' and ψ''.*

Definition 10.2. *Let P be a unification problem. We say that a cascading substitution $[f_1 \mapsto s_1] \circ \cdots \circ [f_n \mapsto s_n]$ for P is a* factor of P *if for every solution σ of P there is a solution σ' for $[f_n \mapsto s_n](\cdots([f_1 \mapsto s_1]P)\cdots)$ such that $\sigma = [f_1 \mapsto s_1] \circ \cdots \circ [f_n \mapsto s_n] \circ \sigma'$. This factor is a* proper *factor if whenever σ and σ' exist as above, $|\sigma'| < |\sigma|$. Otherwise, it is improper.*

Notice that if P is an unsolvable unification problem, then every cascading substitution for P is a factor for P.

Example 10.3. *It is possible for an improper factor ψ to increase the size of a substitution; that is, for $\sigma = \psi \circ \sigma'$ and $|\sigma'| > |\sigma|$ to hold. The unification problem $\forall_a c \exists_a F . F \overset{a}{=} c$ has the single solution $\{(F, c)\}$. It also has the factor $[F \mapsto (H_1 H_2)]$ where*

$$\forall_a c \exists_{a \to a} H_1 \exists_a H_2 \ (H_1 H_2)$$

is $\forall_a c \exists_a F$-free for F. Now

$$\{(F, c)\} = [F \mapsto (H_1 H_2)] \circ \{(H_1, \lambda w.w), (H_2, c)\}$$

but $|\{(H_1, \lambda w.w), (H_2, c)\}| = 2$ and $|\{(F, c)\}| = 1$.

Without exception, we shall be interested only in improper factors that keep the size of solutions constant. Such improper factors always exist for unification problems containing existential quantifiers in their prefix. For example, let x be existentially bound with type $\tau_1 \to \ldots \to \tau_n \to \tau_0$ where $n \geq 0$. If π is a permutation of $\{1, \ldots, n\}$, the substitution $[x \mapsto \lambda w_1 \ldots \lambda w_n . H w_{\pi 1} \ldots w_{\pi n}]$ is an improper factor. This is a kind of renaming operation: x is renamed to H with the possibilities that its arguments are permuted. We shall find improper factor useful since it is possible that applying such a factor to a unification problem yields a problem that is syntactically simplier although its solutions are not "smaller".

Proposition 10.4. *Let ψ' be a factor of P and let ψ'' be a factor of $\psi' P$. Then $\psi' \circ \psi''$ is a factor of P. Furthermore, if both ψ' and ψ'' are proper factors then so is $\psi' \circ \psi''$.*

Proof. Let σ be a solution to P. Then there is a σ' such that σ' solves $\psi' P$ and $\sigma = \psi' \circ \sigma'$. Since ψ'' is a factor of $\psi' P$, then there exists a solution σ'' of $\psi''(\psi' P)$ such that $\sigma' = \psi'' \circ \sigma''$. Thus, $\sigma = (\psi' \circ \psi'') \circ \sigma''$ and σ'' solves $(\psi' \circ \psi'')P$. Hence, $(\psi' \circ \psi'')$ is a factor. If ψ' and ψ'' are proper, then $|\sigma'| < |\sigma|$ and $|\sigma''| < |\sigma'|$, so $|\sigma''| < |\sigma|$ and $\psi' \circ \psi''$ is proper. □

The following proposition shows that factors identified for a subset of the equations in a unification problem are factors for the full unification problem.

Proposition 10.5. *Let P and P' be two unification problems with the same prefixes and be such that the disagreement pairs of P are all contained in P'. It follows that any factor of P must be a factor of P'.*

Proof. Let ψ be a factor of P and let σ be a solution of P'. Thus, σ is a solution of P and $\sigma = \psi \circ \sigma'$ for some σ' where σ' is a solution to ψP. Thus, σ' is also a solution to $\psi P'$. □

We shall now present four different classes of factors: the first two classes are proper and are presented in Propositions 10.6 and 10.8. The third and fourth classes, presented in the next section, are improper factors that can be used to *prune* existential variables of functional types; that is, such variables are replaced with existential variables of fewer arguments. The first class of proper factors is provided as an immediate corollary of Proposition 7.11.

Proposition 10.6. *Let P be a unification problem that contains a flexible-rigid pair with x as the flexible head. If the set of match terms for this disagreement pair is a singleton set, say $\{s\}$, then $[x \mapsto s]$ is a proper factor of P.*

The second class of proper factors arises by generalizing on the usual definition of variable-term disagreement pairs found in first-order unification.

Definition 10.7. *Let P be a unification problem with prefix $\mathcal{Q}$. A* variable defining disagreement pair *is a pair of the form $xy_1 \ldots y_n = t$ where $n \geq 0$, x is an existentially quantified in $\mathcal{Q}$ and does not occur free in t, the variables $y_1, \ldots, y_n$ are distinct and universally quantified to the right of x, and $\lambda y_1 \ldots \lambda y_n t$ is $\mathcal{Q}$-free for x.*

Notice that a variable defining disagreement pair could be flexible-rigid as well as flexible-flexible.

Proposition 10.8. *Let P be unification problem with the variable defining disagreement pair $xy_1 \ldots y_n = t$. The substitution $[x \mapsto \lambda y_1 \ldots \lambda y_n t]$ is a proper factor of P.*

Proof. Let σ be a solution for P. Let s be σx and let $\sigma' := \sigma / \{(x, s)\}$. Using β and η-conversion, we can assume that s is of the form $\lambda y_1 \ldots \lambda y_n s'$, where s' is $\beta\eta$-convertible to σt which is also equal to $\sigma' t$ since x is not free in t. Hence, $[x \mapsto \lambda y_1 \ldots \lambda y_n t] \circ \sigma' = \{(x, \lambda y_1 \ldots \lambda y_n \sigma' t)\} \cup \sigma' = \sigma$. Thus, $[x \mapsto \lambda y_1 \ldots \lambda y_n t]$ is a factor that is also proper since $|\sigma'| = |\sigma| - 1 - |s|$. □

Example 10.9. *Consider the equation*

$$\lambda y_1 \ldots \lambda y_n . X y_{\pi 1} \ldots y_{\pi n} = t$$

where π is a permutation of $\{1,\ldots,n\}$, none of the variables $X, y_1, \ldots, y_n$ are free in t, and the prefix to this equation is of the form $\forall\exists$. Then the substitution

$$[X \mapsto \lambda y_{\pi 1} \ldots \lambda y_{\pi n}.ty_1 \ldots y_n]$$

is a factor.

The substitution provided in the above proposition is the first instance of a substitution that replaces an existential variable with a term that may contain another existential variable in the current unification problem. In all other substitutions so far, substitution terms contained either universal variables or new existential variables.

11. Pruning Unification Problems

We now consider two classes of improper factors we collectively call *pruning factors.* In each case, an existential variable, say x, of functional type is replaced by a new existential variable, say h, of one fewer arguments. This is done using the substitution

$$\psi = [x \mapsto \lambda w_1 \ldots \lambda w_n(hw_1 \ldots w_{i-1}w_{i+1} \ldots w_n)]$$

where $n \geq 0$ and $1 \geq i \geq n$. If ψ is a factor, then it must be an improper factor: let σ and σ' be substitutions such that $\sigma = \psi \circ \sigma'$. There must be some term u and some substitution σ'' that contains neither x nor h in its domain and is such that σ is $\{(x, \lambda w_1 \ldots \lambda w_n\ u)\} \cup \sigma''$ and σ' is $\{(h, \lambda w_1 \ldots \lambda w_{i-1} \lambda w_{i+1} \ldots \lambda w_n\ u)\} \cup \sigma''$. Since the only difference in these substitutions is in the number of abstractions in substitution terms, and since these are not counted by $|\ |$, $|\sigma| = |\sigma'|$.

Proposition 11.1. *Let P be a unification problem with a disagreement pair $xt_1 \ldots t_n = s$, where x is existentially bound with type $\tau_1 \to \cdots \to \tau_n \to \tau_0$ and where $\tau_0, \ldots, \tau_n$ are primitive types. If there is a universal variable y that has a permanent occurrence in t_i, for some $i = 1, \ldots, n$, and that does not have a possible occurrence in s, then*

$$\psi = [x \mapsto \lambda w_1 \ldots \lambda w_n(hw_1 \ldots w_{i-1}w_{i+1} \ldots w_n)]$$

is an improper factor for P.

Proof. Let $xt_1 \ldots t_n = s$ be as in the theorem, with the universal variable y occurring permanently in t_i but not having a possible occurrence in s. Let σ be a solution for P. Since σs cannot contain y free, y cannot be free in $\sigma(xt_1 \ldots t_n)$. Since y occurs permanently in t_i, it also occurs permanently in σt_i. Now σx is η-convertible to a λ-term of the form $\lambda w_1 \ldots \lambda w_n\ u$ where the abstracted variables are of primitive type. If w_i is free in u then $\sigma(xt_1 \ldots t_n)$ would contain σt_i as a subterm and y would be free in $\sigma(xt_1 \ldots t_n)$, which is not possible. Thus, $\sigma = \psi \circ \sigma'$ where σ' is

$$\{(h, \lambda w_1 \ldots \lambda w_{i-1} \lambda w_{i+1} \ldots \lambda w_n\ u)\} \cup \{(v, \sigma(v)) \mid v \in dom(\sigma) \text{ and } v \neq x\}.$$

Hence, ψ is a factor. By the discussion above, it must be an improper factor. □

Example 11.2. *The unification problem*

$$\exists_{i\to i}F\exists_{i\to i}G\forall_i x\forall_i y[Fx \overset{i}{=} Gy]$$

has $[F \mapsto \lambda w\ H_1]$ as a pruning substitution. A pruned form of this unification problem is therefore $\exists H_1\exists G\forall x\forall y[H_1 = Gy]$. This has $[G \mapsto \lambda w\ H_2]$ as a pruning substitution.

A pruned form of this problem is then $\exists H_1 \exists H_2 \forall x \forall y[H_1 = H_2]$*. Thus, the compound substitution,* $[F \mapsto \lambda w\, H_1] \circ [G \mapsto \lambda w\, H_2]$ *is a factor of the original unification problem.*

For another example, consider the unification problem

$$\forall_{i\to i\to i} c \exists_{i\to i\to i} \exists_{i\to i\to i} G \forall_i x \forall_i y [G(cxy)(cyy) \overset{i}{=} Fyy].$$

This problem has a pruning substitution, namely $[G \mapsto \lambda w_1 \lambda w_2 (H w_2)]$*. The resulting unification problem is the simpler* $\forall c \exists F \exists H \forall x \forall y [H(cyy) = Fyy]$*.*

The restriction on the type of the variable being prune is necessary as the unification problem

$$\forall_i a \exists_{(i\to i)\to i\to i} X \forall_i y [X(\lambda z.a) y \overset{i}{=} a]$$

shows. Here, y *has a permanent occurrence in the second argument of* X *while it does not have a possible occurrence in the term* a*. It is not the case, however, that the substitution* $[X \mapsto \lambda w_1 \lambda w_2 (H w_1)]$ *is a factor for this unification. In particular, the solution* $[X \mapsto \lambda z_1 \lambda z_2 (z_1 z_2)]$ *is not in the image of that substitution.*

The type information in a prefix can be used to suggest pruning substitutions. For example, consider the prefix $\mathcal{Q}_1 \exists_{\delta\to\tau} f \mathcal{Q}_2$. A solution for a unification problem with this prefix must provide f with a term of type $\delta \to \tau$ whose free variables are all universally quantified in $\mathcal{Q}_1$. As describe in Section 3, it is possible to use the type information in $\mathcal{Q}_1$ to determine if there is a term of type $\delta \to \tau$. This same type information can be used to determine more about the possible substitution terms for f. For example, it is possible to determine whether there is a term, say $\lambda x.t$, of type $\delta \to \tau$ in which x is free in t. If such a term does not exist, then the pruning substitution $[f \mapsto \lambda w.h]$ is a factor.

In order to determine this structural property given the type information in a prefix, we modify the proof system in Definition 3.5 to enforce that a particular hypothesis is used in a proof.

Definition 11.3. *Let* Δ *be a finite set of types including the type* δ*. The relation* $\Delta \vdash_I^\delta \tau$*, for type* τ*, is defined by the following clauses.*

(1) $\Delta \vdash_I^\delta \delta$.
(2) $\Delta \vdash_I^\delta \tau_1 \to \tau_2$ *if* $\{\tau_1\} \cup \Delta \vdash_I^\delta \tau_2$.
(3) $\Delta \vdash_I^\delta \tau$ *if* δ *is* $\delta_1 \to \cdots \to \delta_n \to \tau$ *and* $\Delta \vdash_I \delta_1, \ldots, \Delta \vdash_I \delta_n$ *(*$n \geq 0$*).*
(4) $\Delta \vdash_I^\delta \tau$ *if* $\delta_1 \to \cdots \to \delta_n \to \tau \in \Delta$ *and* $\Delta \vdash_I \delta_1, \ldots, \Delta \vdash_I \delta_n$ *and for some* $i = 1, \ldots, n$, $\Delta \vdash_I^\delta \delta_i$ *(*$n \geq 1$*).*

Proposition 11.4. *The relationship* $\Delta \vdash_I^\delta \tau$ *is decidable.*

Proof. Provability of $\Delta \vdash_I^\delta \tau$ can only depend on the provability of statements of the form $\Delta' \vdash_I^\delta \tau'$ or $\Delta' \vdash_I \tau'$ where τ' is a subexpression of some type in $\{\tau, \delta\} \cup \Delta$ and Δ' is some finite set of similar types. In particular, there are only a finite number of such values for Δ' and τ', so the length of any proof for $\Delta \vdash_I^\delta \tau$ can be bounded prior to looking for a proof. □

Proposition 11.5. *Let* Δ *be the set of types for quantified variables in the prefix* $\mathcal{Q}$*. There is a* β*-normal* $\mathcal{Q}$*-term* $\lambda x.t$ *of type* $\delta \to \tau$ *such that* x *is free in* t *if and only if* $\Delta \vdash_I^\delta \tau$*.*

PROOF. Assume that $\Delta \vdash_I^\delta \tau$. We show by induction on the length of a proof of this fact that if $\mathcal{Q}\forall_\delta x$ is a prefix whose quantifiers are typed by types in Δ, then there is a $\mathcal{Q}\forall_\delta x$-term of type τ that contains x free. If δ is τ then the required term is simply x. Assume that τ is $\tau_1 \to \tau_2$ and that $\{\tau_1\} \cup \Delta \vdash_I^\delta \tau_2$. By the inductive hypothesis, there is a $\mathcal{Q}\forall_{\tau_1} y \forall_\delta x$-term t' of type τ_2 such that x is free in t'. Thus, $\lambda y.t'$ is the required $\mathcal{Q}\forall_\delta x$-term of type τ which contains x free. Assume that δ is $\delta_1 \to \cdots \to \delta_n \to \tau$ and that $\Delta \vdash_I \delta_1, \ldots, \Delta \vdash_I \delta_n$ $(n \geq 0)$. By Proposition 3.6, there are $\mathcal{Q}\forall_\delta x$-terms, t_i, of type δ_i for $i = 1, \ldots, n$. Thus, $(xt_1 \ldots t_n)$ is the required $\mathcal{Q}\forall_\delta x$-term of type τ which contains x free. Finally, assume that $\delta_1 \to \cdots \to \delta_n \to \tau \in \Delta$ and $\Delta \vdash_I \delta_1, \ldots, \Delta \vdash_I \delta_n$ and for some $i = 1, \ldots, n$, $\Delta \vdash_I^\delta \delta_i$ $(n \geq 0)$. Again, there are $\mathcal{Q}\forall_\delta x$-terms, t_i, of type δ_i for $i = 1, \ldots, n$, one of which contains x free. Thus, if c is bound in $\mathcal{Q}\forall_\delta x$ at type $\delta_1 \to \cdots \to \delta_n \to \tau$ then $(ct_1 \ldots t_n)$ is the required $\mathcal{Q}\forall_\delta x$-term of type τ which contains x free.

Let $\lambda x.t$ be a β-normal $\mathcal{Q}$-term of type $\delta \to \tau$ such that x is free in t. We show by induction on the structure of t that if Δ is the set of types in $\mathcal{Q}$, then $\Delta \vdash_I^\delta \tau$. If t is a variable, then t is x, and we have $\Delta \vdash_I^\delta \delta$ by rule (1). If t is $\lambda y.t'$, then y is different from x and τ is $\tau_1 \to \tau_2$. Thus $\lambda x.t'$ is a β-normal $\mathcal{Q}\forall_{\tau_1} y$-term of type $\delta \to \tau_2$. The inductive hypothesis, $\{\tau_1\} \cup \Delta \vdash_I^\delta \tau_2$, and rule (2) completes this case. Finally, assume that t is of the form $(ct_1 \ldots t_n)$ for c a variable and for some $n > 0$. We now have two cases: either c is x or x appears free in t_i for some $i = 1, \ldots, n$. In the first case, δ is of the form $\delta_1 \to \cdots \to \delta_n \to \tau$ where t_i is a $\mathcal{Q}$-term of type δ_i. By Proposition 3.6 and rule (3), the conclusion follows. Finally, assume that c is not x. Thus x is free in t_i for some $i = 1, \ldots, n$ and c is quantified in $\mathcal{Q}$ with type $\delta_1 \to \cdots \to \delta_n \to \tau$, and the type of t_i is δ_i. This case is completed by the inductive hypothesis, Proposition 3.6, and rule (4) above. □

The following Proposition is now an immediate consequence.

PROPOSITION 11.6. *Let P be a unification problem with prefix $\mathcal{Q}_1 \exists_{\delta \to \tau} f \mathcal{Q}_2$ and let Δ be the set of types that include δ and all the types of quantified variables in $\mathcal{Q}_1$. If $\Delta \vdash_I^\delta \tau$ is not provable then $[f \mapsto \lambda w.H]$ is a factor for P, where H is a variable not in $\mathcal{Q}$.*

EXAMPLE 11.7. *Let list, int, bool, and bt be four primitive types. Let $\mathcal{Q}$ be the prefix that contains universal quantifiers for the following pairs of variables and types:*

$true,$	$bool$	$false,$	$bool$
$zero,$	int	$succ,$	$int \to int$
$node,$	$int \to bt \to bt \to bt$	$leaf,$	$int \to bt$
$sign,$	$int \to bool$	$nil,$	$list$
$bcons,$	$bool \to list \to list$	$icons,$	$int \to list \to list$

Notice that all primitive types and all types built using these primitive types are nonempty. Let Δ be the set of types displayed above. The chart below describes when the relation $\Delta \vdash_I^\delta \tau$ holds given that τ and δ range over the primitive types above. There is a "Y" in the row δ and column τ if $\Delta \vdash_I^\delta \tau$ can be proved; otherwise there is an "N".

	list	*int*	*bool*	*bt*
list	*Y*	*N*	*N*	*N*
int	*Y*	*Y*	*Y*	*Y*
bool	*Y*	*N*	*Y*	*N*
bt	*N*	*N*	*N*	*Y*

Thus, if a unification problem has the prefix $\mathcal{Q}\exists f \mathcal{Q}'$ where the type for f is list → int → bool → bt, this chart can be used to infer that $[f \mapsto \lambda w_1 \lambda w_2 \lambda w_3 . h w_2]$ is a factor.

12. A Subcase of Unification

As an illustration of using some of the technical devices defined in the previous sections, we investigate a class of unification problems in which the search for a pre-unifier give rise to no important choices. That is, if such unification problems have solutions, they have factors that rewrite them into flexible-flexible unification problems. This class contains all first-order unification problems: factors of such problems correspond to most general unifiers. It also contains the class of raised first-order unification problems described in Section 6.

In this section we shall assume that prefixes are restricted to have low orders. In particular, prefixes are restricted so that the type of existentially quantified variables are of order 1 or 0 and universally quantified variables to the right of some existential variable are of primitive type.

Definition 12.1. *A β-normal prefixed term $\mathcal{Q}t$ of type τ is an* argument-restricted *term if the order of τ is 0 or 1 and every subterm of t of the form $x t_1 \ldots t_n$ $(n \geq 0)$ where x is existentially quantified in $\mathcal{Q}$ is such that the terms $t_1, \ldots, t_n$ have as heads distinct variables that are either universally quantified in $\mathcal{Q}$ to the right of x or are internally λ-abstracted variables. A unification problem is argument-restricted if it is argument-restricted as a prefixed term.*

Notice that any unification problem comprised of only first-order terms is trivially an argument-restricted unification problem. This set of unification problems also has useful closure properties.

Proposition 12.2. *Let P be an argument-restricted unification problem. Let P' be the result of Skolemizing, raising, simplifying, pruning, or applying match terms to P. Then P' is argument-restricted.*

Proof. If P' results from Skolemizing P, P' is argument-restricted since Skolemization only introduces new occurrences of existentially quantified variable occurrences which are not applied to any arguments. Since simplifying does not introduce any occurrences of existentially quantified variables, the result of simplifying an argument restricted prefixed term is also argument restricted.

If P' results from raising P, then various subterms of P of the form $x t_1 \ldots t_n$ are replaced with subterms of the form $h y t_1 \ldots t_n$ where y, which is to the left of x in P and is not at the head of any term in $t_1, \ldots, t_n$, is to the right of h in P'. The heads of the terms $y, t_1, \ldots, t_n$ are thus all distinct. Also since y is of primitive type, the type of h is of order 1.

Since pruning has the effect of tossing out arguments to existentially quantified variables, if P' results from pruning P, then P' is clearly argument-restricted.

The case of verifying the application of *match* terms is slightly more involved. Let $\mathcal{Q}t$ be an argument-restricted prefixed term where the prefix is of the form $\mathcal{Q}_1 \exists z \mathcal{Q}_2$ and let s be a match term for z. We show by induction on $|t|$ that the β-normal form of $[z \mapsto s]\mathcal{Q}t$ is a variable-restricted prefixed term. Let $H_1, \ldots, H_m$ be the free variables of s that are not bound in $\mathcal{Q}$ and let $\mathcal{Q}'$ be the prefix $\mathcal{Q}_1\exists H_1\ldots\exists H_m\mathcal{Q}_2$. Thus $[z\mapsto s]\mathcal{Q}t$ is equal to $\mathcal{Q}'[s/z]t$. If $|t|=0$ then t is a variable and $[z\mapsto s]\mathcal{Q}t$ is either $\mathcal{Q}'t$ or $\mathcal{Q}'s$. In either case, the result is argument-restricted.

Now assume that $|t|>0$. Thus, t is of the form $\lambda u_1\ldots\lambda u_p(ae_1\ldots e_n)$ where $p\geq 0$, $n>0$, a is a variable, and the variables $u_1,\ldots,u_p$ are not free in s. Let $a',e'_1,\ldots,e'_n$ be the β-normal forms of $[z\mapsto s]a,[z\mapsto s]e_1,\ldots,[z\mapsto s]e_n$, respectively. By the inductive hypothesis, these terms are argument-restricted. If a is a variable other than z, then the β-normal form of $[z\mapsto s]\mathcal{Q}t$ is $\mathcal{Q}'\lambda u_1\ldots\lambda u_p(ae'_1\ldots e'_n)$, which is argument-restricted. If a is the variable z then $e'_1,\ldots,e'_n$ all have distinct variable heads that are either universally quantified in $\mathcal{Q}$ to the right of z, are in the set $\{u_1,\ldots,u_p\}$, or are bound internally. If s is the imitation term, which is of the form $\lambda w_1\ldots\lambda w_n\ k(H_1w_1\ldots w_n)\ldots(H_mw_1\ldots w_n)$, then the β-normal form of $[z\mapsto s]\mathcal{Q}t$ is

$$\mathcal{Q}'\lambda u_1\ldots\lambda u_pk(H_1e'_1\ldots e'_n)\ldots(H_me'_1\ldots e'_n).$$

Since the terms $e'_1\ldots e'_n$ all have the necessary distinct heads, the term $[z\mapsto s]\mathcal{Q}t$ must also be argument-restricted. If s is a projection term then it must have the form $\lambda w_1\ldots\lambda w_n\ w_i$ for some $i=1,\ldots,n$ since the type of z is at most order 1. Thus, $[z\mapsto s]\mathcal{Q}t$ reduces to $\mathcal{Q}'\lambda u_1\ldots\lambda u_p.e'_i$, which is argument-restricted. □

Example 12.3. *The restrictions on the order of types in the definition of argument-restricted are necessary to achieve closure under the substitution of projection terms. For example, consider the prefixed term*

$$\mathcal{Q}\exists Y\exists X\forall a\forall f.Y(\lambda w(f(Xaw)))a,$$

where $\mathcal{Q}$ is some prefix and the types declared for $Y,X,a,$ and f are $(i\to i)\to i\to i, i\to i\to i, i,$ and $i\to i$, respectively. This prefixed term is not argument-restriction only because the type of Y is of order 2. Applying the projection term $\lambda w_1\lambda w_2.w_1(Hw_1w_2)$ for Y yields the prefixed term

$$\mathcal{Q}\exists H\exists X\forall a\forall f.f(Xa(H(\lambda w(f(Xaw)))a))),$$

which is not argument-restricted for the additional reason that the variable X has an argument that has the existentially quantified variable H as its head.

Our main theorem about this class of unification problems follows from this lemma.

Lemma 12.4. *Let P be a simplified, argument-restricted unification problem. If P is neither a simple failure problem nor a flexible-flexible problem then there exists a proper factor ψ such that ψP is an argument-restricted unification problem.*

Proof. Since P is neither a simple failure problem nor a flexible-flexible problem, it must contain a flexible-rigid pair, say $xt_1\ldots t_n=ks_1\ldots s_m$, with x being the existentially quantified variable. Since P is not a simple failure, the set of match terms for this equation is nonempty. Although this set may not be a singleton, at most one substitution term in this set can lead to a solution. To see this, consider the variable k. If k is universally bound to the left of x in $\mathcal{Q}$, then the imitation term is the only match term that can lead to a solution. If k is universally bound to the right of x in $\mathcal{Q}$, then the only possible match terms leading to a solution are the projection terms. Such projection terms would move one of the arguments, say t_i, into the head position. In this case, the head of t_i

must be the same as k. By assumption, however, there is exactly one such term t_i since all the terms $t_1, \ldots, t_n$ have distinct rigid heads. Thus, the projection term that maps the i^{th} argument of x to the head position must be a factor. In either case, let s be this distinguished match term. The substitutions $[x \mapsto s]$ is the desired factor. □

Theorem 12.5. *Let P be an argument-restricted unification problem that is not flexible-flexible. If a solution to P exists, there exists a proper factor ψ of P such that ψP is a flexible-flexible unification problem.*

Proof. Let P be a simplified, argument-restricted unification problem with solution σ. Also assume that P is not flexible-flexible. Hence, P is not a simple failure problem. We proceed by induction on $|\sigma|$. By Lemma 12.4, P has a proper factor, say ψ, and ψP is an argument-restricted unification problem. If ψP is a flexible-flexible unification problem then we are finished. Otherwise, ψP has a solution σ' such that $\sigma = \psi \circ \sigma'$ and $|\sigma'| < |\sigma|$. By the inductive hypothesis, ψP has a factor ψ' such that $\psi'(\psi P)$ is a flexible-flexible unification problem. The desired proper factor is $\psi \circ \psi'$. □

The subset of argument-restricted unification problems in which existentially quantified variables are applied to terms that are distinct variables bound either internally or to the right of the existential variable is an interesting class of problems. This subset is called L_λ-unification in (Miller, 1991b) and appears to be the weakest extension to first-order unification that treats bound variables via the equations of $\beta\eta$-conversion. Unification in L_λ is decidable and most general unifiers exist when unifiers exist. Unlike above where unification needed to be restricted to essentially second-order, L_λ is ω-ordered.

13. Related Work

For a summary of the early work done on the unification of simply typed λ-terms, see (Huet, 1975). Section 7 is largely a modification of the part of (Huet, 1975) that deals with unification modulo $\beta\eta$-conversion. Snyder & Gallier (1989) present the material of (Huet, 1975) in terms of transformations on sets of equations and provide new completeness proofs. Several abstract properties of unification problems are developed in (Statman, 1981). A declarative specification of simply typed λ-term unification is given in (Miller, 1991a) using a logic programming language as the vehicle for specifying search and λ-term syntax.

The earliest applications of $\beta\eta$-unification were in automating some aspects of deduction in higher-order logic. Theorem proving procedures that incorporated such unification are described in (Andrews *et al.*, 1984; Huet, 1973b; Jensen & Pietrzykowski, 1976; Pietrzykowski, 1971; Pietrzykowski & Jensen, 1976). Also, certain problems regarding the length of proofs can be formulated as unification problems (Farmer, 1984).

A higher-order extension to Horn clauses incorporating $\beta\eta$-unification of simply typed λ-terms is described in (Nadathur, 1987; Nadathur & Miller, 1990) and used as the foundations for an extension of Prolog, called λProlog (Nadathur & Miller, 1988). The Elf logic programming language (Pfenning, 1991) employs $\beta\eta$-unification of dependent-typed λ-terms (Elliott, 1989; Pym, 1990).

The unification of simply typed λ-terms has occasionally been adopted as a component of programs that must manipulate logical expressions or other programs. Huet and Lang

(1978) demonstrated that second-order matching, a decidable subset of λ-term unification, could be used to analysis and transform programs in a natural fashion. Their ideas were enriched by moving their analysis into λProlog (Hannan & Miller, 1988; Hannan & Miller, 1989; Miller & Nadathur, 1987). In the domain of the manipulation of logical expressions, λ-term unification has been used to implement inference rules in the Isabelle theorem prover (Paulson, 1986; 1989) and in theorem provers written in λProlog (Felty & Miller, 1988; Felty, 1989). Elliott and Pfenning (1988) further extended this notion and refer to the use of λ-terms and unification to analyze logical expressions and programs as *higher-order abstract syntax*.

Acknowledgements. I am grateful to Peter Andrews, Masami Hagiya, Sunil Issar, Gopalan Nadathur, Larry Paulson, Frank Pfenning, David Wolfram, and the anonymous referees for useful comments on an earlier drafts of this paper. This work was supported in part by the grants ONR N00014-88-K-0633, NSF CCR-87-05596, and DARPA N00014-85-K-0018, and by a British Science and Engineering Research Council visiting fellowship research grant.

References

Andrews, P. (1971). Resolution in Type Theory. *Journal of Symbolic Logic* **36**, 414 – 432.

Andrews, P. (1972). General Models and Extensionality. *Journal of Symbolic Logic* **37**, 395 – 397.

Andrews, P., Cohn, E., Miller, D., Pfenning, F. (1984). Automating higher order logic. In *Automated Theorem Proving: After 25 Years*, eds. W. Bledsoe and D. Loveland, American Mathematical Society, Providence, RI, 169 – 192.

Andrews, P. (1986). *An Introduction to Mathematical Logic and Type Theory.* Academic Press, 1986.

Church, A. (1940). A formulation of the simple theory of types. *Journal of Symbolic Logic* **5**, 56 – 68.

Elliott, C., Pfenning, F. (1988). Higher-Order Abstract Syntax. *Proceedings of the ACM-SIGPLAN Conference on Programming Language Design and Implementation*, 199 – 208, ACM Press.

Elliott, C. (1989). Higher-Order Unification with Dependent Types. *Rewriting Techniques and Applications*, Springer-Verlag LNCS **355**, 121 – 136.

Farmer. W. (1984). *Length of Proofs and Unification Theory.* PhD. dissertation, University of Wisconsin, Madison.

Felty, A., Miller, D. (1988). Specifying Theorem Provers in a Higher-Order Logic Programming Language. *Proceedings of the Ninth International Conference on Automated Deduction, Argonne, IL*, eds. E. Lusk and R. Overbeek, Springer-Verlag LNCS **310**, 61 – 80.

Felty, A. (1989). *Specifying and Implementing Theorem Provers in a Higher-Order Logic Programming Language.* PhD dissertation, University of Pennsylvania.

Goldfarb, W. (1981). The Undecidability of the Second-Order Unification Problem. *Theoretical Computer Science* **13**, 225 – 230.

Hannan, J., Miller, D. (1988). Uses of Higher-Order Unification for Implementing Program Transformers. *Fifth International Conference and Symposium on Logic Programming*, ed. K. Bowen and R. Kowalski, MIT Press, 942 – 959.

Hannan, J., Miller, D. (1989). A Meta Language for Functional Programs. *Meta-Programming in Logic Programming*, eds. H. Rogers and H. Abramson, MIT Press.

Henkin, L. (1950). Completeness in the theory of types. *Journal of Symbolic Logic* **15**, 81 – 91.

Hindley, J., Seldin, J. (1986). *Introduction to Combinators and λ-calculus.* Cambridge University Press.

Huet, G. (1973a). The Undecidability of Unification in Third Order Logic. Information and Control **22**, 257 – 267.

Huet, G. (1973b). A Mechanization of Type Theory. *Proceedings of the Third International Joint Conference on Artificial Intelligence*, 139 – 146.

Huet, G. (1975). A Unification Algorithm for Typed λ-Calculus. *Theoretical Computer Science* **1**, 27 – 57.

Huet, G., Lang, B. (1978). Proving and Applying Program Transformations Expressed with Second-Order Patterns. *Acta Informatica* **11**, 31 – 55.

Jensen, D., Pietrzykowski, T. (1976). Mechanizing ω-Order Type Theory Through Unification. *Theoretical Computer Science* **3**, 123 – 171.

Lucchesi, C. (1972). The Undecidability of the Unification Problem for Third Order Languages. Report CSRR 2059, Dept. of Applied Analysis and Computer Science, University of Waterloo.

Miller, D. (1987). A Compact Representation of Proofs. *Studia Logica* **46/4**, 345 – 368.

Miller, D. (1991a). Unification of Simply Typed Lambda-Terms as Logic Programming. *Eight International Logic Programming Conference*, Paris, ed. K. Furukawa, MIT Press, 255 – 269.

Miller, D. (1991b). A Logic Programming Language with Lambda-Abstraction, Function Variables, and Simple Unification. *Journal of Logic and Computation* **2/4**, 497 – 536.

Miller, D., Nadathur, G. (1987). A Logic Programming Approach to Manipulating Formulas and Programs. *Symposium on Logic Programming*, San Franciso, ed. S. Haridi, IEEE Press, 379 – 388.

Nadathur, G. (1987). *A Higher-Order Logic as the Basis for Logic Programming.* PhD dissertation, University of Pennsylvania.

Nadathur, G., Miller, D. (1988). An Overview of λProlog. Fifth International Conference on Logic Programming, MIT Press, 810 – 827.

Nadathur, G., Miller, D. (1990). Higher-order Horn Clauses. *Journal of the ACM*, **37/4**, 777 – 814.

Paulson, L. (1986). Natural Deduction as Higher-Order Resolution. *Journal of Logic Programming* **3/3**, 237 – 258.

Paulson, L. (1989). The Foundation of a Generic Theorem Prover. *Journal of Automated Reasoning*, **5**, 363 – 397.

Pfenning, F. (1988). Partial Polymorphic Type Inference and Higher-Order Unification. *Proceedings of the 1988 ACM Conference on Lisp and Functional Programming.*

Pfenning, F. (1991). Logic Programming in the LF Logical Framework. In *Logical Frameworks*, eds. G. Huet and G. Plotkin, Cambridge University Press, 149 – 181.

Pietrzykowski, T. (1971). A Complete Mechanization of Second-Order Logic. *Journal of the ACM* **20/2**, 333 – 364.

Pietrzykowski, T., Jensen, D. (1976). Mechanizing ω-Order Type Theory Through Unification. *Theoretical Computer Science* **3**, 123 – 171.

Pym, D. (1990). Proofs, Search and Computation in General Logic. PhD dissertation, University of Edinburgh.

Snyder, W., Gallier, J. (1989). Higher Order Unification Revisited: Complete Sets of Transformations. *Journal of Symbolic Computation*, **8/1 − 2**, 101 – 140.

Statman, R. (1979). Intuitionistic Propositional Logic is Polynomial-Space Complete. *Theoretical Computer Science* **9**, 67 – 72.

Statman, R. (1981). On the Existence of Closed Terms in the Typed λ-Calculus II: Transformations of Unification Problems. *Theoretical Computer Science* **15**, 329 – 338.