UNIFICATION PROCEDURES IN AUTOMATED DEDUCTION METHODS BASED ON MATINGS: A SURVEY

Jean Gallier

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

Note: This research was conducted while the author was on sabbatical at Digital PRL

January 26, 2001

This research was partially supported by ONR under Grant No. N00014-88-K-0593.

UNIFICATION PROCEDURES IN AUTOMATED DEDUCTION METHODS BASED ON MATINGS: A SURVEY

Jean Gallier

Abstract: Unification procedures arising in methods for automated theorem proving based on matings are surveyed. We begin by reviewing some fundamentals of automated deduction, including the Skolem form and the Skolem-Herbrand-Gödel theorem. Next, the method of matings for first-order languages without equality due to Andrews and Bibel is presented. Standard unification is described in terms of transformations on systems (following the approach of Martelli and Montanari, anticipated by Herbrand). Some fast unification algorithms are also sketched, in particular, a unification closure algorithm inspired by Paterson and Wegman's method. The method of matings is then extended to languages with equality. This extention leads naturally to a generalization of standard unification called rigid E-unification (due to Gallier, Narendran, Plaisted, and Snyder). The main properties of rigid E-unification, decidability, NP-completeness, and finiteness of complete sets, are discussed.

Unification Procedures In Automated Deduction Methods Based on Matings: A Survey

Jean H. Gallier
Digital PRL, and
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104

1 Introduction

Unification is a very general computational paradigm that plays an important role in many different areas of symbolic computation. For example, unification plays a central role in

- Automated Deduction (First-order logic with or without equality, higher-order logic);
- Logic Programming (Prolog, λ-Prolog);
- Constraint-based Programming;
- Type Inferencing (ML, ML$^+$, etc.);
- Knowledge-Base Systems, Feature structures; and
- Computational Linguistics (Unification grammars).

In this survey, we shall focus on unification problems arising in methods for automated theorem proving based on matings. This covers at least the kind of unification arising in resolution (Robinson [61], Plotkin [58]), matings (Andrews, Bibel [4, 11, 12, 13]), equational matings (Gallier, Plaisted Narendran, Raatz, Snyder [28, 27]), and ET-proofs (Miller, Pfenning [57, 51, 53]). Clearly, many other important parts of unification theory are left out, and we apologize for this. In particular, we will not cover the classification theory of the Siekmann school (for example, [62, 69, 15, 16]), the many unification procedures for special theories (AC, etc., see Siekmann [62]), the combination of unification procedures (for example, Yelick [71], Schmidt-Schauss [65], Boudet, Jouannaud, and Schmidt-Schauss [14]) order-sorted unification (for example, Meseguer, Goguen, and Smolka [50], and Isakowitz [37]), semi-unification (see [41] for references), unification applied to type-inferencing (for example, Milner [54], Kfoury, Tiuryn, and Urzyczyn, [40,42], and Remy [59, 60]), unification in computational linguistics (for example, Shieber [63]), and unification in feature structures (for example, Aït-Kaci [2, 3]). Fortunately for the uninitiated reader, there are other very good survey papers covering significant parts of the above topics: the survey by Siekmann [62], the survey by Knight [43], and the survey by Kirchner and Jouannaud [38]. The most elementary and having the broadest coverage is probably Knight's survey. Our account has a narrower focus, but it also sketches some of the proof techniques, which is usually missing from the other surveys. The topics that we will cover are:

- Standard unification;
- Rigid E-unification (E a finite set of first-order equations), a decidable form of E-unification recently introduced in the framework of Andrews and Bibel's method of matings [4, 6, 11, 12, 13].

These unification problems will be tackled using the *method of transformations* on term systems, already anticipated in Herbrand's thesis [32] (1930), and revived very effectively by Martelli and Montanari [49] for standard unification. In a nutshell, the method is as follows:

A unification problem is gradually transformed into one whose solution is (almost) obvious.

This approach is an instance of a very old method in mathematics, but a fairly recent trend in computer science, namely, the specification of procedures and algorithms in terms of inference rules. There are a number of significant advantages to this method. (1) A clean separation of logic and control is achieved. (2) The correctness of the procedure obtained this way is often easier to establish, and irrelevant implementation issues are avoided. (3) The actual design of algorithms from the procedure specified by rules can be viewed as an optimization process.

Another benefit of this approach to the design of algorithms is that one often gains a deeper understanding of the problem being solved, and one understands more easily the differences between algorithms solving a same problem. The effectiveness of this method for tackling unification problems was first shown by Martelli and Montanari [49] (although, as we said earlier, it was anticipated by Herbrand [32]). Similarly, Bachmair, Dershowitz, Hsiang, and Plaisted [8, 7, 9, 10] showed how to describe and study Knuth-Bendix completion procedures [44] in terms of proof rules. Presented in terms of proof rules, completion procedures are more transparent, and their correctness proofs are significantly simplified. Many other examples of the effectiveness of the method of proof rules can be easily found (for example, in type inference problems, and Gentzen-style automated deduction, see Gallier [22] for the latter). Jouannaud and Kirchner [38] also emphasize the proof rules method, and provide many more examples of its use.

Although in this paper the perspective is to discuss unification in terms of transformations (proof rules), we should not forget about the history of unification theory, and the major turning points.[1] Undoubtedly, the invention of the resolution method and of the first unification algorithm by Alan Robinson in the early sixties [61] (1965) marks the beginning of a new era. In the post Robinson era, we encounter Plotkin's seminal paper on building-in equational theories [58] (1972), in which E-unification is introduced, and then Gérard Huet's thesis [35] (1976) (with Huet [34] as a precursor). Huet's thesis (1976) makes a major contribution to the theory of higher-order unification, in that it shows that a restricted form of unification, preunification, is sufficient for most theorem-proving applications. The first practical higher-order (pre)unification algorithm is defined and proved correct. Huet also gives a quasi-linear unification algorithm for standard unification, and an algorithm for unifying infinite rational trees. In the more recent past, in our perspective, we would like to mention Martelli and Montanari's paper showing the effectiveness of the method of transformations [49] (1982), and Claude Kirchner's thesis [39] in which the method of transformations is systematically applied to E-unification. We also would like to mention that most of the results on E-unification and higher-order unification discussed in this paper originate from Wayne Snyder's thesis [66] (1988). A more comprehensive presentation of these results will appear in Snyder [68].

Unification theory is a very active field of research, and it is virtually impossible to keep track of all the papers that have appeared on this subject. As evidence that unification is a very active field of research, two special issues of the *Journal of Symbolic Computation* are devoted to unification theory (Part I in Vol. 7(3 & 4), and Part II in Vol. 8(1 & 2), both published in 1989). It is our hope that this paper will inspire other researchers to work in this area.

The paper is organized as follows. Section 2 provides a review of background material relevant to unification. The notion of Skolem form and the Skolem-Herbrand-Gödel theorem are reviewed in Section 3. The method of matings for languages without equality is presented in Section 4. Section 5 is devoted to a presentation of standard unification using the method of transformations on terms systems. Fast unification methods are discussed in Section 6. The method of equational matings, a generalization of matings to languages with equality, is presented in Section 7. Section 8 is devoted to rigid E-unification, an extension of standard unification arising in the framework of equational matings. We give an overview of results of Gallier, Narendran, Plaisted, and Snyder [27], showing among other things that rigid E-unification is decidable and NP-complete. Directions for further research are discussed in section 9.

[1] The following list is by no means exclusive, and only reflects the perspective on unification adopted in this paper.

2 Algebraic Background

We begin with a brief review of algebraic background material. The purpose of this section is to establish the notation and the terminology used throughout this paper. As much as possible, we follow Huet [36] and Gallier [22].

Definition 2.1 Let $\longrightarrow \subseteq A \times A$ be a binary relation on a set A. The *converse* (or *inverse*) of the relation $\longrightarrow$ is the relation denoted as $\longrightarrow^{-1}$ or $\longleftarrow$, defined such that $u \longleftarrow v$ iff $v \longrightarrow u$. The symmetric closure of $\longrightarrow$, denoted by $\longleftrightarrow$, is the relation $\longrightarrow \cup \longleftarrow$. The transitive closure, reflexive and transitive closure, and the reflexive, symmetric, and transitive closure of $\longrightarrow$ are denoted respectively by $\stackrel{+}{\longrightarrow}$, $\stackrel{*}{\longrightarrow}$, and $\stackrel{*}{\longleftrightarrow}$.

Definition 2.2 A relation $\longrightarrow$ on a set A is *Noetherian* or *well founded* iff there are no infinite sequences $\langle a_0, \ldots, a_n, a_{n+1}, \ldots \rangle$ of elements in A such that $a_n \longrightarrow a_{n+1}$ for all $n \geq 0$.

Definition 2.3 A *preorder* $\preceq$ on a set A is a binary relation $\preceq \subseteq A \times A$ that is reflexive and transitive. A *partial order* $\preceq$ on a set A is a preorder that is also antisymmetric. The converse of a preorder (or partial order) $\preceq$ is denoted as $\succeq$. A *strict ordering* (or *strict order*) $\prec$ on a set A is a transitive and irreflexive relation. Given a preorder (or partial order) $\preceq$ on a set A, the strict ordering $\prec$ associated with $\preceq$ is defined such that $s \prec t$ iff $s \preceq t$ and $t \not\preceq s$. Conversely, given a strict ordering $\prec$, the partial ordering $\preceq$ associated with $\prec$ is defined such that $s \preceq t$ iff $s \prec t$ or $s = t$. The converse of a strict ordering $\prec$ is denoted as $\succ$. Given a preorder (or partial order) $\preceq$, we say that $\preceq$ is well founded iff $\succ$ is well founded.

Definition 2.4 Let $\longrightarrow \subseteq A \times A$ be a binary relation on a set A. We say that $\longrightarrow$ is *locally confluent* iff for all $a, a_1, a_2 \in A$, if $a \longrightarrow a_1$ and $a \longrightarrow a_2$, then there is some $a_3 \in A$ such that $a_1 \stackrel{*}{\longrightarrow} a_3$ and $a_2 \stackrel{*}{\longrightarrow} a_3$. We say that $\longrightarrow$ is *confluent* iff for all $a, a_1, a_2 \in A$, if $a \stackrel{*}{\longrightarrow} a_1$ and $a \stackrel{*}{\longrightarrow} a_2$, then there is some $a_3 \in A$ such that $a_1 \stackrel{*}{\longrightarrow} a_3$ and $a_2 \stackrel{*}{\longrightarrow} a_3$. We say that $\longrightarrow$ is *Church-Rosser* iff for all $a_1, a_2 \in A$, if $a_1 \stackrel{*}{\longleftrightarrow} a_2$, then there is some $a_3 \in A$ such that $a_1 \stackrel{*}{\longrightarrow} a_3$ and $a_2 \stackrel{*}{\longrightarrow} a_3$. We say that $a \in A$ is *irreducible* iff there is no $b \in A$ such that $a \longrightarrow b$. It is well known (Huet [36]) that a Noetherian relation is confluent iff it is locally confluent and that a relation is confluent iff it is Church-Rosser. A relation $\longrightarrow$ is *canonical* iff it is Noetherian and confluent. Given a canonical relation $\longrightarrow$, it is well known that every $a \in A$ reduces to a unique irreducible element $a{\downarrow} \in A$ called the *normal form of* a, and that $a \stackrel{*}{\longleftrightarrow} b$ iff $a{\downarrow} = b{\downarrow}$ (Huet [36]).

Definition 2.5 Terms are built up inductively from a *ranked alphabet* (or *signature*) Σ of constant and function symbols, and a countably infinite set $\mathcal{X}$ of *variables*. For simplicity of exposition, we assume that Σ is a one-sorted ranked alphabet, i.e., that there is a *rank function* $r\colon \Sigma \to \mathbf{N}$ assigning a *rank* (or *arity*) $r(f)$ to every symbol $f \in \Sigma$ ($\mathbf{N}$ denotes the set of natural numbers). We let $\Sigma_n = \{f \in \Sigma \mid r(f) = n\}$. Symbols in Σ_0 (of rank zero) are called *constants*.

Definition 2.6 We let $T_\Sigma(\mathcal{X})$ denote the set of terms built up inductively from Σ and $\mathcal{X}$. Thus, $T_\Sigma(\mathcal{X})$ is the smallest set with the following properties:

- $x \in T_\Sigma(\mathcal{X})$, for every $x \in \mathcal{X}$;
- $c \in T_\Sigma(\mathcal{X})$, for every $c \in \Sigma_0$;
- $f(t_1, \ldots, t_n) \in T_\Sigma(\mathcal{X})$, for every $f \in \Sigma_n$ and all $t_1, \ldots, t_n \in T_\Sigma(\mathcal{X})$.

Given a term $t \in T_\Sigma(\mathcal{X})$, we let $Var(t)$ be the set of variables occurring in t. A term t is a *ground term* iff $Var(t) = \emptyset$.

It is well known that $T_\Sigma(\mathcal{X})$ is the term algebra freely generated by $\mathcal{X}$, and this allows us to define substitutions.

Definition 2.7 A *substitution* is a function $\varphi\colon \mathcal{X} \to T_\Sigma(\mathcal{X})$ such that $\varphi(x) \neq x$ for only finitely many $x \in \mathcal{X}$. The set $D(\varphi) = \{x \in \mathcal{X} \mid \varphi(x) \neq x\}$ is the *domain* of φ, and the set $I(\varphi) = \bigcup_{x \in D(\varphi)} Var(\varphi(x))$ is the *set of variables introduced* by φ.

A substitution $\varphi\colon \mathcal{X} \to T_\Sigma(\mathcal{X})$ with domain $D(\varphi) = \{x_1, \ldots, x_n\}$ and such that $\varphi(x_i) = t_i$ for $i = 1, \ldots, n$, is denoted as $[t_1/x_1, \ldots, t_n/x_n]$. Since $T_\Sigma(\mathcal{X})$ is freely generated by $\mathcal{X}$, every substitution $\varphi\colon \mathcal{X} \to T_\Sigma(\mathcal{X})$ has a unique homomorphic extension $\widehat{\varphi}\colon T_\Sigma(\mathcal{X}) \to T_\Sigma(\mathcal{X})$. For every term $t \in T_\Sigma(\mathcal{X})$, we denote $\widehat{\varphi}(t)$ as $t[\varphi]$ or even as $\varphi(t)$ (with an intentional identification of φ and $\widehat{\varphi}$).

Definition 2.8 Given two substitutions φ and ψ, their *composition* denoted $\varphi\,;\psi$ is the substitution defined such that $\varphi\,;\psi(x) = \widehat{\psi}(\varphi(x))$ for all $x \in \mathcal{X}$. Thus, note that $\varphi\,;\psi = \varphi \circ \widehat{\psi}$, but not $\varphi \circ \psi$, where $\circ$ denotes the composition of functions (written in diagram order). A substitution φ is *idempotent* iff $\varphi\,;\varphi = \varphi$. It is easily seen that a substitution φ is idempotent iff $I(\varphi) \cap D(\varphi) = \emptyset$. A substitution φ is a *renaming* iff $\varphi(x)$ is a variable for every $x \in D(\varphi)$, and φ is injective over its domain. Given a set V of variables and a substitution φ, the *restriction of* φ *to* V is the substitution denoted $\varphi|_V$ defined such that, $\varphi|_V(x) = \varphi(x)$ for all $x \in V$, and $\varphi|_V(x) = x$ for all $x \notin V$.

There will be occasions where it is necessary to replace a subterm of a given term with another term. We can make this operation precise by defining the concept of a tree address originally due to Gorn.

Definition 2.9 Given a term $t \in T_\Sigma(\mathcal{X})$, the set $Tadd(t)$ of *tree addresses* in t is a set of strings of positive natural numbers defined as follows (where ϵ denotes the null string):

- $Tadd(x) = \{\epsilon\}$, for every $x \in \mathcal{X}$;
- $Tadd(c) = \{\epsilon\}$, for every $c \in \Sigma_0$;
- $Tadd(f(t_1, \ldots, t_n)) = \{\epsilon\} \cup \{iw \mid w \in Tadd(t_i),\ 1 \leq i \leq n\}$.

Definition 2.10 Given any $\beta \in Tadd(t)$, the *subtree* rooted at β in t is denoted as t/β. Given $t_1, t_2 \in T_\Sigma(\mathcal{X})$ and $\beta \in Tadd(t_1)$, the tree $t_1[\beta \leftarrow t_2]$ obtained by replacing the subtree rooted at β in t_1 with t_2 can be easily defined.

Definition 2.11 Let $\longrightarrow$ be a binary relation $\longrightarrow\ \subseteq T_\Sigma(\mathcal{X}) \times T_\Sigma(\mathcal{X})$. (i) The relation $\longrightarrow$ is *monotonic* (or *stable under the algebra structure*) iff for every two terms s, t and every function symbol $f \in \Sigma$, if $s \longrightarrow t$ then $f(\ldots, s, \ldots) \longrightarrow f(\ldots, t, \ldots)$.

(ii) The relation $\longrightarrow$ is *stable* (under substitution) if $s \longrightarrow t$ implies $s[\sigma] \longrightarrow t[\sigma]$ for every substitution σ.

Definition 2.12 A strict ordering $\prec$ has the *subterm property* iff $s \prec f(\ldots, s, \ldots)$ for every term $f(\ldots, s, \ldots)$. A *simplification ordering* $\prec$ is a strict ordering that is monotonic and has the subterm property (since we are considering symbols having a fixed rank, the deletion property is superfluous, as noted in Dershowitz [20]). A *reduction ordering* $\prec$ is a strict ordering that is monotonic, stable (under substitution), and such that $\succ$ is well founded. With a slight abuse of language, we will also say that the converse $\succ$ of a strict ordering $\prec$ is a simplification ordering (or a reduction ordering). It is shown in Dershowitz [20] that there are simplification orderings that are total on ground terms.

Definition 2.13 A *set of rewrite rules* is a binary relation $R \subseteq T_\Sigma(\mathcal{X}) \times T_\Sigma(\mathcal{X})$ such that $Var(r) \subseteq Var(l)$ whenever $\langle l, r \rangle \in R$. A rewrite rule $\langle l, r \rangle \in R$ is usually denoted as $l \rightarrow r$. A rewrite rule $s \rightarrow t$ is a *variant* of a rewrite rule $u \rightarrow v \in R$ iff there is some renaming ρ with domain $Var(u) \cup Var(v)$ such that $s = u[\rho]$ and $t = v[\rho]$.

Let $R \subseteq T_\Sigma(\mathcal{X}) \times T_\Sigma(\mathcal{X})$ be a set of rewrite rules.

Definition 2.14 The relation $\longrightarrow_R$ over $T_\Sigma(\mathcal{X})$ is defined as the smallest stable and monotonic relation that contains R. This is the *rewrite relation* associated with R. This relation is defined explicitly as follows: Given any two terms $t_1, t_2 \in T_\Sigma(\mathcal{X})$, then

$$t_1 \longrightarrow_R t_2$$

iff there is some variant $l \rightarrow r$ of some rule in R, some tree address β in t_1, and some substitution σ, such that

$$t_1/\beta = l[\sigma], \quad \text{and} \quad t_2 = t_1[\beta \leftarrow r[\sigma]].$$

The concept of an equation is similar to that of a rewrite rule, but equations can be used oriented forward or backward, and the restriction $Var(r) \subseteq Var(l)$ is dropped.

Definition 2.15 A *set of equations* is a binary relation $E \subseteq T_\Sigma(\mathcal{X}) \times T_\Sigma(\mathcal{X})$. An equation $\langle l, r \rangle \in E$ is usually denoted as $l \doteq r$, to emphasize the difference with a rewrite rule. An equation $s \doteq t$ is a *variant* of an equation $u \doteq v \in E$ iff there is some renaming ρ with domain $Var(u) \cup Var(v)$ such that $s = u[\rho]$ and $t = v[\rho]$.

Definition 2.16 The relation $\longleftrightarrow_E$ over $T_\Sigma(\mathcal{X})$ is defined as the smallest symmetric relation containing E that is stable, and monotonic. This relation is defined explicitly as follows: Given any two terms $t_1, t_2 \in T_\Sigma(\mathcal{X})$, then

$$t_1 \longleftrightarrow_E t_2$$

iff there is some variant $l \doteq r$ of some equation in $E \cup E^{-1}$, some tree address β in t_1, and some substitution σ, such that

$$t_1/\beta = l[\sigma], \quad \text{and} \quad t_2 = t_1[\beta \leftarrow r[\sigma]].$$

Note that an equation can be used oriented forward or backward, since E^{-1} consists of all $r \doteq l$ such that $l \doteq r \in E$.

Definition 2.17 The reflexive and transitive closure of $\longrightarrow_R$ is denoted as $\overset{*}{\longrightarrow}_R$, and the reflexive and transitive closure of $\longleftrightarrow_E$ as $\overset{*}{\longleftrightarrow}_E$. Sometimes, $\overset{*}{\longleftrightarrow}_E$ is denoted as $=_E$. It is easily seen that $\overset{*}{\longleftrightarrow}_E$ is an equivalence relation. In fact, $\overset{*}{\longleftrightarrow}_E$ is the smallest congruence containing E that is stable under substitution, and $\overset{*}{\longleftrightarrow}_E \; = (\longrightarrow_E \cup \longrightarrow_E^{-1})^*$. It can be shown that $E \models u \doteq v$ iff $u \overset{*}{\longleftrightarrow}_E v$ (a form of Birkhoff's completeness theorem). A set R of rewrite rules is called Noetherian, confluent, Church-Rosser, or canonical, iff the relation $\longrightarrow_R$ has the corresponding property.

3 The Skolem-Herbrand-Gödel Theorem

An automated deduction method is a procedure for checking whether an arbitrary formula is provable. In this survey, we are restricting ourselves to first-order logic, for which, by Gödel's completeness theorem, we know that a formula is provable iff it is valid. Thus, the problem is equivalent to designing a procedure for checking whether an arbitrary formula is valid. This problem is also called the *validity problem*. The main difficulty in automated deduction is to deal with the quantifiers. To be more precise, the difficulty is to design procedures that handle the quantifiers efficiently. For a quantifier-free formula, also called a proposition, it can be shown that the validity problem is decidable. For propositions without equality, this is fairly easy to show, and there are a number of algorithmic methods for deciding the validity problem: the truth-table method, the resolution method, the matings method, Davis and Putnam's method, etc. (see Gallier [22], or Manna [48]). For quantifier-free formulae with equality, the validity problem is also decidable, but this is harder to prove. Decidability can be established using the "congruence closure method", and an algorithm using congruence closure can be designed (Kozen [45,46], Nelson and Oppen [55], Downey, Sethi, and Tarjan [21]). In the general case of quantifed formulae, Church [18] (1936) proved that the validity problem is undecidable. A particularly simple proof of this important result was later given by Floyd (see Manna [48], page 105-106).

Two important theorems help in dealing with quantifiers. The first theorem, essentially due to Skolem, shows that the validity problem can be reduced to the validity problem for formulae containing only one kind of quantifiers (say $\forall$). The second one, known as the Skolem-Herbrand-Gödel theorem, shows that the validity of a quantified formula can be reduced to the validity of a *quantifier-free formula*, modulo guessing some (ground) substitutions. Without digressing excessively, we would like to warn the readers of a confusion often made between two different important theorems, Herbrand's theorem, and the Skolem-Herbrand-Gödel theorem. The first theorem, *Herbrand's theorem* [32] (1930), is about the *provability* of first-order formulae, and its (meta)proof does not appeal to the semantics of first-order logic at all. Furthermore, Herbrand's theorem also yields some information on the length of proofs. Historically, Herbrand's theorem was proved in 1930, before the Skolem-Herbrand-Gödel theorem (which, according to Peter Andrews, was apparently only formulated in the early fifties by Quine). The second theorem, the *Skolem-Herbrand-Gödel theorem* (see Andrews [4, 5] or Gallier [22]), is about *unsatisfiability*, a semantic notion, and its (meta)proof can be presented essentially as a semantic argument (a certain model is constructed). Furthermore, the Skolem-Herbrand-Gödel theorem does not yield any information on the length of proofs. Basically, the Skolem-Herbrand-Gödel theorem combines results of Skolem [64] (1928) and Gödel [30, 31] (1930), from his proof of the completeness theorem, but since it is definitely a "semantic version" of Herbrand's theorem, it is appropriate to refer to it by the concatenation of the three names! Finally, the (meta)proof of Herbrand's theorem is significantly harder than the (meta)proof of the Skolem-Herbrand-Gödel theorem, but it yields more information, namely, some complexity-theoretic information about the length of proofs. For our purposes, the Skolem-Herbrand-Gödel theorem is all we need.

What Skolem (essentially) showed is that given a quantified (first-order) formula A, one can associate two formulae A_{vff} and A_{sff} with the following properties:

(1) The formula A_{vff} contains only *existensial* quantifiers, and A is valid iff A_{vff} is valid.

(2) The formula A_{sff} contains only *universal* quantifiers, and A is satisfiable iff A_{sff} is satisfiable.

Following Goldfarb (see [32]), we call A_{vff} the *validity functional form* of A. Intuitively speaking, it is obtained from A by "eliminating" universal quantifiers. Dually, we call A_{sff} the *satisfiability functional form* of A. Intuitively speaking, it is obtained from A by "eliminating" existential quantifiers. Now, recall that we say that a formula A is unsatisfiable iff it is not satisfied in any structure, and that A is valid iff $\neg A$ is unsatisfiable. Thus, (2) can be equivalently stated as A is unsatisfiable iff A_{sff} is unsatisfiable.

Oddly, early researchers in the field of automated deduction have shown a preference for A_{sff}, the *satisfiability functional form*, often referred to as the *Skolem form* of A. This is perhaps because the main property of A_{sff} (A is satisfiable iff A_{sff} is satisfiable) can be intuitively justified by an appeal to the axiom of choice. The consequence of this bias is that most automated deduction methods are traditionally presented as *refutation* methods. This means that in order to show that A is valid, we attempt to show that $\neg A$ is unsatisfiable, that is, we try to show that the Skolem form $(\neg A)_{\mathit{sff}}$ of $\neg A$, is unsatisfiable. We have an unfortunate first step which consists in negating what we are trying to prove! We believe that tradition is worth fighting when it is silly, and it would certainly make more sense to present automated deduction methods, the resolution method in particular, in their positive version, as *proving* methods, rather than as *refutation* (negative) methods. The drawback of such a choice is that one has to constantly translate traditional refutation methods into their positive form, in order to compare them with other (positive) proving methods. Consequently, mostly for ease of comparison with other methods, we will follow the tradition, not without some guilt feelings, and present our methods as refutation methods. Therefore, we will be using the *satisfiability functional form* A_{sff}, often called the *Skolem form* of A, and use the version of the Skolem-Herbrand-Gödel theorem dealing with unsatisfiability, rather than validity.

Roughly, the Skolem-Herbrand-Gödel theorem asserts that a formula A is unsatisfiable iff some conjunction of substitution instances of subformulae of the Skolem form of A is unsatisfiable. The crucial idea is that the unsatisfiability of a *quantified* formula A is reduced to the unsatisfiability of some *quantifier-free* formula (obtainable from A). The price of this reduction is that one needs to "guess" some substitutions and which subformulae of A need to be instantiated with these substitutions.

It is possible to define the Skolem form of an arbitrary formula and state a version of the Skolem-Herbrand-Gödel theorem for arbitrary formulae. However, in the fully general case, one needs to deal with negative and positive occurrences of subformulae, which complicates matters and obscures the main point of the theorem. We can give a simpler version of the Skolem-Herbrand-Gödel theorem for formulae in a special form, the *negation normal form* (for short, *nnf*), where negation only applies to atomic formulae. Since every formula is equivalent to another formula in *nnf*, there is no loss of generality. Furthermore, contrary to other normal formal forms, such as prenex form, the *nnf* of a formula is linear in the size of the original formula. The following example should give a crisper idea of what we are talking about.

Example 3.1 Let $A = \exists x \forall y (P(y) \supset P(x))$. In order to prove that A is valid, we will attempt to prove that $\neg A$ is unsatisfiable.

Step 1: Compute $\neg A$. We have

$$\neg A = \forall x \exists y (P(y) \wedge \neg P(x)).$$

Step 2: Compute the Skolem form $\forall x B_0$ of $\neg A$. We have

$$\forall x B_0 = \forall x (P(f(x)) \wedge \neg P(x)).$$

Step 3: Find a conjunction of (ground) instances of B_0 which is unsatisfiable. Observe that

$$C = (P(f(a)) \wedge \neg P(a)) \wedge (P(f(f(a))) \wedge \neg P(f(a)))$$

is unsatisfiable.

One should note that no substitution σ makes $\sigma(B_0) = \sigma(P(f(x)) \wedge \neg P(x))$ unsatisfiable. A systematic way to find a conjunction of (ground) instances of B_0 which is unsatisfiable is to duplicate B_0 and try again. After duplication (with renaming of the second conjunct), we have

$$B_1 = (P(f(x)) \wedge \neg P(x)) \wedge (P(f(y)) \wedge \neg P(y)).$$

The key point is that unsatisfiability will be achieved if we can find **mated pairs** of literals, that is, pairs of literals of opposite signs. In B_1, the literals $P(f(x))$ and $\neg P(y)$ form a mated pair. If we apply the substitution $[a/x,\ f(a)/y]$, we get $P(f(a))$ and $\neg P(f(a))$. What happens is that $P(f(x))$ and $P(y)$ are **unified** by the substitution $[a/x,\ f(a)/y]$.

This is a general phenomenon, and it is at the heart of the method of matings of Bibel and Andrews [4, 6, 11, 12, 13]. The crucial observation due to Andrews and Bibel is that a quantifier-free formula (in *nnf*) is unsatisfiable iff certain sets of literals occurring in A (called *vertical paths*) are unsatisfiable. Matings come up as a convenient method for checking that vertical paths are unsatisfiable. Roughly speaking, a *mating* is a set of pairs of literals of opposite signs (mated pairs) such that all these (unsigned) pairs are globally unified by some substitution. The importance of matings stems from the fact that a quantifier-free formula A has a mating iff there is a ground substitution θ such that $\theta(A)$ is unsatisfiable. Thus, we see where unification comes into the picture, at least in the case of formulae without equality. Things are more complicated when formulae contain equality. In this case, we are naturally led to more general forms of unification, E-unification and rigid E-unification. We now proceed with a more rigorous presentation of the concept of Skolem form and of the Skolem-Herbrand-Gödel theorem. First, we recall the formal definition of the negation normal form.

Definition 3.2 Formulae in *negation normal form* (for short, in *nnf*) are defined inductively as follows. A formula A is in *nnf* iff either

(1) A is an atomic formula or the negation $\neg B$ of an atomic formula, or

(2) $A = (B \vee C)$, where B and C are in *nnf*, or

(3) $A = (B \wedge C)$, where B and C are in *nnf*, or

(4) $A = \forall x B$, where B is in *nnf*, or

(5) $A = \exists x B$, where B is in *nnf*.

Lemma 3.3 *For every formula A, one can construct a formula B in nnf such that $A \equiv B$ is valid.*

From now on, we will be dealing only with *rectified formulae*, that is, formulae in which no variable occurs both free and bound, and distinct occurrences of quantifiers bound distinct variables. It is easy to show that for every formula A, one can construct a rectified formula B equivalent to A. We now give an algorithm to compute the Skolem form of a formula in *nnf*. First, it is necessary to compute the universal scope of a subformula.

Definition 3.4 Given a (rectified) formula A in *nnf*, the set $US(A)$ of pairs $\langle B, L\rangle$ where B is a subformula of A and L is a sequence of variables, is defined inductively as follows:

$$
\begin{aligned}
US_0 &= \{\langle A, \langle\rangle\rangle\};\\
US_{k+1} &= US_k \cup \{\langle C, L\rangle, \langle D, L\rangle \mid \langle B, L\rangle \in US_k,\\
&\qquad B \text{ is of the form } (C \wedge D) \text{ or } (C \vee D)\}\\
&\qquad \cup \{\langle C, L\rangle \mid \langle \exists x C, L\rangle \in US_k\}\\
&\qquad \cup \{\langle C, \langle y_1, \ldots, y_m, x\rangle\rangle \mid \langle \forall x C, \langle y_1, \ldots, y_m\rangle\rangle \in US_k\}.
\end{aligned}
$$

For every subformula B of A, the sequence L of variables such that $\langle B, L\rangle$ belongs to $US(A) = \bigcup US_k$ is the *universal scope* of B.

Example 3.5 Let

$$A = \forall x(P(a) \vee \exists y(Q(y) \wedge \forall z(P(y,z) \vee \exists u Q(x,u)))) \vee \exists w Q(a,w).$$

Then,

$$
\begin{gathered}
\langle \exists y(Q(y) \wedge \forall z(P(y,z) \vee \exists u Q(x,u))), \langle x\rangle\rangle,\\
\langle \exists u Q(x,u), \langle x, z\rangle\rangle, \text{ and}\\
\langle \exists w Q(a,w), \langle\rangle\rangle
\end{gathered}
$$

define the universal scope of the subformulae of A of the form $\exists x B$.

Definition 3.6 Given a rectified sentence[2] A in *nnf*, the *Skolem form* (or *Skolem normal form*) of A is defined recursively as follows. Let A' be any subformula of A:

(i) If A' is either an atomic formula B or the negation $\neg B$ of an atomic formula B, then $SK(A') = A'$.

(ii) If A' is of the form $(B * C)$, where $* \in \{\vee, \wedge\}$, then $SK(A') = (SK(B) * SK(C))$.

(iii) If A' is of the form $\forall x B$, then $SK(A') = \forall x SK(B)$.

(iv) If A' is of the form $\exists x B$, then if $\langle y_1, \ldots, y_m\rangle$ is the universal scope of $\exists x B$ (that is, the sequence of variables such that $\langle \exists x B, \langle y_1, \ldots, y_m\rangle\rangle \in US(A)$) then

  (a) If $m > 0$, create a new *Skolem function symbol* $f_{A'}$ of rank m and let $SK(A') = SK(B[f_{A'}(y_1, \ldots, y_m)/x])$.

  (b) If $m = 0$, create a new *Skolem constant* $f_{A'}$ and let $SK(A') = SK(B[f_{A'}/x])$.

Observe that since the sentence A is rectified, all subformulae A' of the form $\exists x B$ are distinct, and since the Skolem symbols are indexed by the subformulae A', they are also distinct.

[2] Recall that a *sentence* is a formula without any free variables, and it is also called a *closed formula*.

Example 3.7 Let

$$A = \forall x(P(a) \lor \exists y(Q(y) \land \forall z(P(y,z) \lor \exists u Q(x,u)))) \lor \exists w Q(a,w).$$

$SK(\exists w Q(a,w)) = Q(a,c)$,
$SK(\exists u Q(x,u)) = Q(x, f(x,z))$,
$SK(\exists y(Q(y) \land \forall z(P(y,z) \lor \exists u Q(x,u)))) =$

$$(Q(g(x)) \land \forall z(P(g(x),z) \lor Q(x,f(x,z)))), \quad \text{and}$$

$SK(A) = \forall x(P(a) \lor (Q(g(x)) \land \forall z(P(g(x),z) \lor Q(x,f(x,z))))) \lor Q(a,c)$.

The main property of Skolem forms is given in the following lemma.

Lemma 3.8 *Let* **L** *be a first-order language with or without equality. Let A be a rectified* **L***-sentence in nnf, and let B be its Skolem normal form. The sentence A is satisfiable iff its Skolem form B is satisfiable.*

Proof. Let C be any subformula of A. We show that the following properties hold:

(a) For every structure **A** such that all function, predicate, and constant symbols in the Skolem form $SK(C)$ of C receive an interpretation, for every assignment s, if $\mathbf{A} \models SK(C)[s]$ then $\mathbf{A} \models C[s]$.

(b) For every structure **A** such that exactly all function, predicate, and constant symbols in C receive an interpretation, for every assignment s (with range A), if $\mathbf{A} \models C[s]$ then there is an expansion **B** of **A** such that $\mathbf{B} \models SK(C)[s]$.

The proof is by induction on the size of subformulae of A. Details can be found in Gallier [22]. $\square$

Warning: In general, a formula A and its Skolem form $SK(A)$ are **not** equivalent. For example, $\exists x P(x)$ and $P(a)$ are not equivalent.

The Skolem-Herbrand-Gödel theorem can be stated in a very concise form if we introduce the notion of a *compound instance* due to Andrews. Recall that a *literal* is either an atomic formula or the negation of an atomic formula.

Definition 3.9 Let A be a rectified sentence in *nnf* and let B its Skolem form. The set of *compound instances* (for short, *c-instances*) of B is defined inductively as follows:

(i) If B is a literal, then B is its only c-instance;

(ii) If B is of the form $(C * D)$, where $* \in \{\lor, \land\}$, for any c-instance H of C and c-instance K of D, then $(H * K)$ is a c-instance of B;

(iii) If B is of the form $\forall x C$, for any k closed terms $t_1, \ldots, t_k$, if H_i is a c-instance of $C[t_i/x]$ for $i = 1, \ldots, k$, then $H_1 \land \ldots \land H_k$ is a c-instance of B.

Example 3.10 Let

$$B = \forall x(P(x) \lor \forall y Q(y, f(x))) \land (\neg P(a) \land (\neg Q(a, f(a)) \lor \neg Q(b, f(a)))).$$

Then,

$$(P(a) \lor (Q(a, f(a)) \land Q(b, f(a)))) \land (\neg P(a) \land (\neg Q(a, f(a)) \lor \neg Q(b, f(a))))$$

is a c-instance of B.

Note that c-instances are quantifier free. We are now ready to state a version of the Skolem-Herbrand-Gödel theorem due to Andrews [4] (1981), but first, a minor technicality has to be taken care of. If the first-order language under consideration does not have any constants, the theorem fails. For example, the formula $\forall x \forall y(P(x) \land \neg P(y))$ is unsatisfiable, but if the language has no constants, we cannot find a ground substitution instance of $P(x) \land \neg P(y)$ that is unsatisfiable. To avoid this problem, we will assume that if any first-order language $\mathbf{L}$ does not have constants, the special constant $\#$ is added to it.

Theorem 3.11 *Let* $\mathbf{L}$ *be a first-order language with or without equality. Given any rectified sentence* A *in nnf, if* B *is the Skolem form of* A*, then* A *is unsatisfiable if and only if some compound instance* C *of* B *is unsatisfiable.*

Remark: There is an algorithm for deciding whether a c-instance is unsatisfiable if equality is absent, but in case equality is present, such an algorithm is much less trivial. Such an algorithm based on congruence closure exists.

In view of lemma 3.8, if we are interested in deciding unsatisfiability, we can restrict our attention to universal sentences (that is, sentences containing only universal quantifiers). Then, theorem 3.11 can be sated as follows:

Corollary 3.12 *Given a universal sentence* A *in nnf,* A *is unsatisfiable if and only if some compound instance* C *of* A *is unsatisfiable.*

Lemma 3.12 is the theoretical basis of many refutation procedures, in particular the resolution method, and the method of matings. In the next section, we look at the method of matings, as presented by Andrews [4]. The same method was also investigated by Bibel [11, 12, 13] under the name of "connection method", and in fact, probably predates the method of matings.

4 The Method of Matings

If one wants to write a procedure based on lemma 3.12, the first problem to solve is to find a way of generating compound instances nicely. Andrews proposed a convenient notion, the notion of *amplification* [4].

Definition 4.1 Given a universal formula A, C is obtained from B by *quantifier duplication* iff C results from B by replacing some subformula $\forall x M$ of B by $(\forall x M \wedge \forall x M)$.

If $C_1 \Rightarrow C_2$, ..., $C_{n-1} \Rightarrow C_n$, with $B = C_1$, $C = C_n$, and C_{i+1} is obtained from C_i by quantifier duplication, $1 \leq i < n$, then C is obtained from B by some *sequence of quantifier duplications*.

If $A \Rightarrow^* B$ by some sequence of quantifier duplications, C is a rectified sentence equivalent to B, and D obtained from C by deleting the quantifiers in C, then D is an *amplification* of A.

The following lemma shows that every compound instance arises from some amplification.

Lemma 4.2 *Let* $\mathbf{L}$ *be a first-order language with or without equality. Given a universal sentence* A *in nnf,* C *is a c-instance of* A *iff there is some amplification* D *of* A *and some (ground) substitution* θ *such that* $C = \theta(D)$.

Form theorem 3.11 and lemma 4.2, we have the following variant of the Skolem-Herbrand-Gödel theorem.

Theorem 4.3 *Let* $\mathbf{L}$ *be a first-order language with or without equality. Given a universal sentence* A *in nnf,* A *is unsatisfiable iff there is some amplification* D *of* A *and some (ground) substitution* σ *such that* $\sigma(D)$ *is unsatisfiable.*

The next step towards automated deduction is to find a method for deciding whether, given a quantifier-free formula D, there is some susbtitution σ such that $\sigma(D)$ is undecidable. We first solve this problem for the case of first-order languages **without equality**. We present the method of *vertical paths*, a variant of the disjunctive normal form.

Definition 4.4 Let A be a quantifier-free formula in *nnf*. The set $vp(A)$ of *vertical paths* in A is the set of sets of literals defined inductively as follows:

If A is a literal, then $vp(A) = \{\{A\}\}$;

If $A = (B \wedge C)$, then $vp(A) = \{\pi_1 \cup \pi_2 \mid \pi_1 \in vp(B), \pi_2 \in vp(C)\}$;

If $A = (B \vee C)$, then $vp(A) = vp(B) \cup vp(C)$.

The fundamental property of vertical paths in given in the following lemma.

Lemma 4.5 *Let* $\mathbf{L}$ *be a first-order language with or without equality. Given a quantifier-free formula* A *in nnf,* A *is unsatisfiable iff every vertical path in* A *is unsatisfiable.*

Let us now look more closely at vertical paths, and see what it means for a vertical path to be unsatisfiable. This is where the assumption that equality does not occur simplifies matters drastically. Given a literal L, if $L = A$ where A is a positive atom, then $\neg L = \neg A$, else if $L = \neg A$ where A is a positive atom, then $\neg L = A$. We say that L and $\neg L$ are *complementary*.

- For languages **without** equality, a vertical path $\{L_1, \ldots, L_m\}$ is unsatisfiable iff two of the literals L_i, L_j are complementary, that is, $L_i = \neg L_j$.
- If the formula A is of the form $\sigma(D)$, this means that there are literals $\sigma(L_i)$ and $\neg\sigma(L_j)$ such that
$$\sigma(L_i) = \sigma(L_j).$$

A substitution such that $\sigma(L_i) = \sigma(L_j)$ is called a *unifier* of L_i and L_j. Thus, we see that looking for an automated deduction procedure based on the Skolem-Herbrand-Gödel theorem leads to unification. It also leads to *matings*, which are convenient for checking that vertical paths are unsatisfiable.

Definition 4.6 Given a quantifier-free formula A in *nnf*, a *mating* for A is a pair $\mathcal{M} = \langle MS, \sigma \rangle$, where

(1) MS is a set of pairs of literals of opposite sign (in A), and

(2) σ is a substitution such that, for every pair $(L, \neg L') \in MS$,

$$\sigma(L) = \sigma(L').$$

A mating is *p-acceptable* iff every vertical path $\pi \in vp(A)$ contains some mated pair $(L, \neg L') \in MS$.

The following lemma is the bridge between theorem 4.3 and lemma 4.5.

Lemma 4.7 *Given a quantifier-free formula* A *in nnf, the following properties hold:*

(1) Given a substitution θ*, if* $\theta(A)$ *is unsatisfiable, then there is a p-acceptable mating* $\mathcal{M}$ *for* A.

(2) If $\mathcal{M}$ is a p-acceptable mating for A with associated substitution $\sigma_{\mathcal{M}}$, then $\sigma_{\mathcal{M}}(A)$ is unsatisfiable.

The completeness and soundness for the method of matings is an immediate consequence of theorem 4.3, lemma 4.5, and lemma 4.7.

Theorem 4.8 *Given a universal sentence A in nnf, A is unsatisfiable iff some amplification D of A has a p-acceptable mating.*

Let us work out an example in detail to illustrate theorem 4.8.

Example 4.9 (Due to Andrews [4]) Let A be the formula

$$\exists x \forall y (Px \equiv Py) \supset (\exists x Px \equiv \forall y Py).$$

We want to prove that A is valid. First, we negate A and eliminate $\equiv$ and $\supset$:

$$\exists x \forall y [(\neg Px \vee Py) \wedge (\neg Py \vee Px)] \wedge [(\exists x Px \wedge \exists y \neg Py) \vee (\forall y Py \wedge \forall x \neg Px)]$$

Next, we skolemize:

$$\forall y [(\neg Pc \vee Py) \wedge (\neg Py \vee Pc)] \wedge [(Pd \wedge \neg Pe) \vee (\forall z Pz \wedge \forall x \neg Px)]$$

Next, we amplify (duplicate quantifiers):

$$\forall y [(\neg Pc \vee Py) \wedge (\neg Py \vee Pc)] \wedge \forall y [(\neg Pc \vee Py) \wedge (\neg Py \vee Pc)] \\ \wedge [(Pd \wedge \neg Pe) \vee (\forall z Pz \wedge \forall x \neg Px)]$$

Rectify variables:

$$\forall y [(\neg Pc \vee Py) \wedge (\neg Py \vee Pc)] \wedge \forall w [(\neg Pc \vee Pw) \wedge (\neg Pw \vee Pc)] \\ \wedge [(Pd \wedge \neg Pe) \vee (\forall z Pz \wedge \forall x \neg Px)]$$

Delete Quantifiers:

$$[(\neg Pc \vee Py) \wedge (\neg Py \vee Pc)] \wedge [(\neg Pc \vee Pw) \wedge (\neg Pw \vee Pc)] \\ \wedge [(Pd \wedge \neg Pe) \vee (Pz \wedge \forall x \neg Px)]$$

Vertical paths displayed in "matrix form":

$$\begin{bmatrix} \neg Pc & \vee & Py \\ \neg Py & \vee & Pc \\ \neg Pc & \vee & Pw \\ \neg Pw & \vee & Pc \\ & & \\ \begin{bmatrix} Pd \\ \neg Pe \end{bmatrix} & \vee & \begin{bmatrix} Pz \\ \neg Px \end{bmatrix} \end{bmatrix}$$

There are 32 vertical paths.

The substitution $\theta = [d/x,\ d/y,\ c/z,\ e/w]$ "mates" all the vertical paths:

$$\begin{bmatrix} \neg Pc & \vee & Pd \\ \neg Pd & \vee & Pc \\ \neg Pc & \vee & Pe \\ \neg Pe & \vee & Pc \\ & & \\ \begin{bmatrix} Pd \\ \neg Pe \end{bmatrix} & \vee & \begin{bmatrix} Pc \\ \neg Pd \end{bmatrix} \end{bmatrix}$$

A p-acceptable mating is given below:

$$\{\langle \neg Pc, Pz\rangle,\ \langle \neg Py, Pd\rangle,\ \langle \neg Pc, Pc\rangle,\ \langle Py, \neg Px\rangle,\ \langle \neg Pe, Pw\rangle\}$$

The substitution θ unifies (in fact, in an *mgu*) of the set of pairs:

$$\{\langle Pc, Pz\rangle,\ \langle Py, Pd\rangle,\ \langle Pc, Pc\rangle,\ \langle Py, Px\rangle,\ \langle Pe, Pw\rangle\}$$

A naive procedure implementing the method of matings is given below.

Definition 4.10 (A Procedure for Finding Matings)

Let A_0 be a universal sentence in *nnf*. The formula A_0 evolves in steps called *quantifier duplication steps*.

Let A be the evolving formula

Let $\widehat{A}$ be obtained from A by deleting the quantifiers (an *amplification* of A_0).

Initially, $A := A_0$.

1. : Construct $vp(\widehat{A})$, the set of sets of literals called *vertical paths*.
2. : Find whether there is a substitution σ such that for every vertical path $\pi \in vp(\widehat{A})$, $\sigma(\pi)$ is unsatisfiable. If step 2 succeeds, go to step 4. Otherwise, go to step 3.
3. : Choose some universal subformula $\forall x B$ of A, and replace it by $(\forall x B \wedge \forall x B)$. Then, rectify variables in this new formula, obtaining A'. Let $A := A'$ (*quantifier duplication step*). Go back to step 1.
4. : Stop, A_0 is unsatisfiable (and so are $\widehat{A}$ and A).

If A_0 is unsatisfiable, this procedure stops when it succeeds in finding some substitution closing all vertical paths in step 2. For languages without equality, we can use *unification* to check whether a set of pairs can be mated.

We now briefly discuss some optimizations of the naive procedure. To trim the search space, we can use a *connection graph* (Kowalski). Given a mating $\mathcal{M}$ for A, literals M and $\neg N$ are *potential mates* w.r.t. $\mathcal{M}$ iff some vertical path contains both M and $\neg N$ and $\sigma_{\mathcal{M}}(M)$ and $\sigma_{\mathcal{M}}(N)$ are unifiable (i.e., there is a substitution such that $\theta(\sigma_{\mathcal{M}}(M)) = \theta(\sigma_{\mathcal{M}}(N))$). The *connection graph* for A and $\mathcal{M}$ has the literals of A as nodes, and there is an edge from M to $\neg N$ iff M and $\neg N$ are potential mates w.r.t. $\mathcal{M}$. When building a mating, we can choose a pair of potential mates and add it to the current mating.

The way of eliminating $\equiv$ can have a great influence on the number of vertical paths. $A \equiv B$ can be transformed to

$$(\neg A \vee B) \wedge (A \vee \neg B), \quad \text{or} \quad (A \wedge B) \vee (\neg A \wedge \neg B).$$

Example 4.11 (revisited) Let A be the formula

$$\exists x \forall y (Px \equiv Py) \supset (\exists x Px \equiv \forall y Py).$$

Negate and eliminate $\equiv$ and $\supset$:

$$\exists x \forall y [(Px \wedge Py) \vee (\neg Px \wedge \neg Py)] \wedge [(\exists x Px \wedge \exists y \neg Py) \vee (\forall y Py \wedge \forall x \neg Px)]$$

Skolemize:

$$\forall y [(Pc \wedge Py) \vee (\neg Pc \wedge \neg Py)] \wedge [(Pd \wedge \neg Pe) \vee (\forall z Pz \wedge \forall x \neg Px)]$$

Duplicate quantifier and rectify:

$$\forall y [(Pc \wedge Py) \vee (\neg Pc \wedge \neg Py)] \wedge \forall w [(Pc \wedge Pw) \vee (\neg Pc \wedge \neg Pw)] \\ \wedge [(Pd \wedge \neg Pe) \vee (\forall z Pz \wedge \forall x \neg Px)]$$

Delete quantifiers, and display in matrix form:

$$\begin{bmatrix} \begin{bmatrix} Pc \\ Py \end{bmatrix} & \vee & \begin{bmatrix} \neg Pc \\ \neg Py \end{bmatrix} \\ \\ \begin{bmatrix} Pc \\ Pw \end{bmatrix} & \vee & \begin{bmatrix} \neg Pc \\ \neg Pw \end{bmatrix} \\ \\ \begin{bmatrix} Pd \\ \neg Pe \end{bmatrix} & \vee & \begin{bmatrix} Pz \\ \neg Px \end{bmatrix} \end{bmatrix}$$

There are 8 vertical paths instead of 32.

We now briefly discuss some ways of reducing the number of vertical paths. Observe that

$$M = \Big[(L_1 \wedge P_1) \vee \ldots \vee (L_n \wedge P_n)\Big] \wedge \Big[(\neg L'_1 \wedge \ldots \wedge \neg L'_n \wedge Q) \vee R\Big]$$

is equivalent to

$$N = \Big[(L_1 \wedge P_1) \vee \ldots \vee (L_n \wedge P_n)\Big] \wedge R,$$

when $L_i = L'_i$.

If A is a sentence containing M and A^* is the result of substituting N for M in A,

$$\begin{bmatrix} \begin{bmatrix} L_1 \\ P_1 \end{bmatrix} \quad \vee \quad \ldots \quad \vee \quad \begin{bmatrix} L_n \\ P_n \end{bmatrix} \\ \\ \begin{bmatrix} \neg L_1 \\ \vdots \\ \neg L_n \\ Q \end{bmatrix} \quad \vee \quad R \end{bmatrix}$$

occurs in A, and

$$\begin{bmatrix} \begin{bmatrix} L_1 \\ P_1 \end{bmatrix} \quad \vee \quad \ldots \quad \vee \quad \begin{bmatrix} L_n \\ P_n \end{bmatrix} \\ \\ R \end{bmatrix}$$

occurs in A^*.

If $\mathcal{M}$ is a mating for A such that each pair $\langle L_i, \neg L'_i\rangle$ or $\langle L'_i, \neg L_i\rangle$ is in $\mathcal{M}$, there is a mating $\mathcal{M}^*$ associated with $\sigma_{\mathcal{M}}(A^*)$.

Another kind of simplification due to Prawitz is as follows: $(L \vee Q) \wedge (\neg L \vee R)$ simplifies to $(L \wedge R) \vee (\neg L \wedge Q)$. Also, we can try to use symmetries in matings to minimize the search space. For more details, the reader is referred to Andrews [4].

5 Standard Unification

We saw in the previous section how (standard) unification arises naturally in the context of the method of matings. Historically, unification was brought to the fore as a seminal component of automated deduction systems by Robinson in 1964, and has been studied by numerous researchers since that time. In this section we present an abstract view of unification as a set of non-deterministic rules for transforming a unification problem into an explicit representation of its solution, if such exists. This elegant approach is due to Martelli and Montanari [49], but was in fact implicit in Herbrand's thesis (1930).[3] For a very good historical account and technical details on standard unification, the reader is referred to Knight's survey article [43], Jouannaud and Kirchner's survey article [38], and Lassez, Maher, and Marriot [47].

It is natural to define a unification problem as a set $\{\langle u_1, v_1\rangle, \ldots, \langle u_n, v_n\rangle\}$ of ordered pairs of terms. This is fine, but it turns out that it is more convenient to allow repetitions of pairs $\langle u_i, v_i\rangle$, and to allow $\langle u_i, v_i\rangle$ to be unordered. Thus, we adopt the following definition of a unification problem.

Definition 5.1 A *term pair* or just a *pair* is a multiset of two terms, denoted, by $\langle u, v\rangle$. A *term system* (or *system*) is a (finite) multiset of term pairs. In denoting term systems, we will often drop the curly brackets and simply write $\langle u_1, v_1\rangle, \ldots, \langle u_n, v_n\rangle$.

The reason why multisets are more convenient than sets is that they lead to a simpler statement of the transformations. This shows up in two ways. Firstly, since multiset union is not idempotent (contrary to set union), when we write $S \cup \{\langle u, v\rangle\}$, we mean the multiset consisting of all pairs in S distinct from $\langle u, v\rangle$, and of the $m+1$ pairs $\langle u, v\rangle$, where m is the number of occurrences of $\langle u, v\rangle$ in S. Thus, in a transformation $S \cup \{\langle u, v\rangle\} \Longrightarrow S \cup R$, where S and R are multisets of (unordered pairs) and $\langle u, v\rangle$ does not belong to R, there are fewer occurrences of the pair $\langle u, v\rangle$ on the right-hand side of the transformations than there are on the left-hand side. If S, R were interpreted as sets, and $\cup$ as set union, we would have to stipulate that $\langle u, v\rangle$ does not belong to S, or explicitly remove it from S on the right-hand side. Secondly, if pairs $\langle u, v\rangle$ are considered ordered, then a special transformation switching pairs $\langle u, x\rangle$ to $\langle x, u\rangle$ is needed when x is a variable but u is not. If we treat $\langle u, v\rangle$ as unordered, such a transformation is unnecessary. We can now state the *(standard) unification problem*:

Definition 5.2 Given a term system $S = \langle u_1, v_1\rangle, \ldots, \langle u_n, v_n\rangle$, the (standard) unification problem is to find some (all) substitution(s) σ s.t. $u_i[\sigma] = v_i[\sigma]$ for every i, $1 \leq i \leq n$.

We let $U(S)$ denote the set of all unifiers of S.

Example 5.3 The substitution $\sigma = [a/x,\ a/y]$ is a unifier of the pair

$$\langle f(x, g(a, y)),\ f(x, g(y, x))\rangle.$$

In fact, σ is the only unifier of this pair.

[3] It is remarkable that in his thesis, Herbrand gave all the steps of a (nondeterministic) unification algorithm based on transformations on systems of equations. These transformations are given at the end of the section on property A, page 148 of Herbrand [32].

Example 5.4 For every term t, the substitution $\sigma = [a/x,\ a/y,\ t/z]$ is a unifier of the pair $\langle f(z, g(a, y)),\ f(z, g(y, x))\rangle$.

Observe that in example 5.4, there is an infinite number of unifiers. This leads us to the following question.

Question: How do we compare unifiers?

Answer: Define a preorder $\leq$ on substitutions.

Definition 5.5 Let V be a set of variables. We write $\sigma = \theta[V]$ iff $\sigma(x) = \theta(x)$ for all $x \in V$, and we write $\sigma \leq \theta[V]$ (*σ is more general than θ over V*) iff there exists a substitution η such that $\theta = \sigma\,;\eta[V]$.

The intuitive idea behind these definitions is that σ is more general than θ (over V) when each $\theta(x)$ can be obtained from $\sigma(x)$ by instantiating some of the variables occurring in $\sigma(x)$. The reason for relativizing the definition of $\leq$ to a set V of variables is technical. For one thing, some of the results are incorrect if V is left out. Also, in theorem proving applications, it is often desirable to compare substitutions with respect to a "protected set" of variables V. A crucial concept in unification theory is that of a *most general unifier*.

Definition 5.6 Given a term system S and a finite set V of "protected" variables, a substitution σ is a *most general unifier for S away from V* (or *mgu away from V*) iff:

(i) $D(\sigma) \subseteq Var(S)$ and $I(\sigma) \cap (V \cup D(\sigma)) = \emptyset$;

(ii) σ is a unifier of S, i.e. $\sigma \in U(S)$;

(iii) For every unifier θ of S, $\sigma \leq \theta[Var(S)]$.

Note that condition (i) implies that σ is idempotent. When V is not significant, we just call σ an *mgu*. A number of questions now arise naturally.

Some Questions:

- 1. Is $\leq$ well-founded?
- 2. Given S, can we decide whether S is unifiable?
- 3. Given S, if S is unifiable, is there a *mgu*?

The answer to all questions is YES.

That $\leq$ is well-founded is shown in Huet [35]. The decidability of standard unification is implicit in Herbrand's thesis [32] (1930), and it is also settled by Robinson [61] (1965), who gives the first algorithm to find *mgu*'s.

We will now show that questions (1)-(3) have a positive answer. A key point is the similarity between solving a unification problem and solving a system of linear equations:

$$\begin{gathered} u_1 = v_1 \\ \vdots \\ u_n = v_n \end{gathered}$$

and

$$\begin{gathered} x_1 = a_{1\,1}x_1 + \cdots + a_{1\,n}x_n \\ \vdots \\ x_m = a_{m\,1}x_1 + \cdots + a_{m\,n}x_n. \end{gathered}$$

However, there are some major differences.

- In unification, we cannot assume that the algebraic structure is a field.
- The u_i may not be variables.
- We may have $u_i = v_j$ for $i \neq j$.

Nevertheless, the basic idea of *variable elimination* (as in Gaussian elimination) applies. If u_i is a variable that does not occur in v_i, we can *substitute* v_i for u_i in the rest of the system, and preserve the set of solutions:

$$\begin{gathered} u_1[v_i/u_i] = v_1[v_i/u_i] \\ \vdots \\ u_i = v_i \\ \vdots \\ u_n[v_i/u_i] = v_n[v_i/u_i]. \end{gathered}$$

This leads to the idea of the *method of transformations* on term systems:

Attempt to transform S into a system S' which is obviously solved.

One of the critical issues is to decide what we mean by a solved system. Quite obviously, a solved system is one that should represent a unifying substitution. Since we are dealing with multisets of unordered pairs, we have to be a little careful in formalizing this idea.

Definition 5.7 A term pair $\langle u, v\rangle$ is in *solved form* in a system S iff either u or v is a variable, say x, and this variable x does not occur anywhere else in S; in particular, if $x = u$ then $x \notin Var(v)$ (and similarly if $x = v$ then $x \notin Var(u)$). The variable x is called a *solved variable*. A system is in solved form if all its pairs are in solved form; a variable is *unsolved* if it occurs in S but is not solved.

Note that a solved form system is always a *set* of solved pairs. Also, note that in a solved pair $\langle u, v\rangle$, it is possible that both u and v are variables, and in this case, it may be that only one of the two is solved in S, or that both are solved in S. Thus, ignoring the order in term pairs, a system is *solved* iff it is of the form

$$S = \langle x_1, v_1\rangle, \ldots, \langle x_n, v_n\rangle,$$

where $x_1, \ldots, x_n$ are **distinct variables**, and $x_i \notin Var(v_j)$ for all i, j, $1 \leq i, j \leq n$. A system in solved form defines essentially a unique substitution as shown in the definition below.

Definition 5.8 Given a system S in solved form, we define the substitution σ_S as follows: if $S = \langle x_1, v_1\rangle, \ldots, \langle x_n, v_n\rangle$, then $\sigma_S = [v_1/x_1, \ldots, v_n/x_n]$.

Actually, the above definition is ambiguous, and this is the one place where we might regret our definition of a term system where pairs $\langle u, v\rangle$ are unordered. Let us explain where the difficulty lies. There is no problem with a solved pair $\langle u, v\rangle$ in which **only one** of u, v, say u, is a solved variable, because then the substitution component must be $[v/u]$. But when **both** u and v are solved variables, we can use $[u/v]$ or $[v/u]$ interchangeably as a substitution component. Thus, σ_S is not uniquely defined. However, note for any two σ'_S and σ''_S obtained from S, there is a renaming permutation ρ such that $\sigma'_S = \sigma''_S\,; \rho$ (where ρ is determined by the pairs $\langle u, v\rangle$ where both u and v are solved in S). Thus, σ_S is uniquely defined, modulo some inessential renaming permutation. The important fact is that σ_S is an idempotent *mgu* of S, as we now show.

Lemma 5.9 *Let $S = \langle x_1, t_1\rangle, \ldots, \langle x_n, t_n\rangle$ be in solved form, where the $x_1, \ldots, x_n$ are solved variables. If $\sigma = [t_1/x_1, \ldots, t_n/x_n]$, then σ is an idempotent mgu of S. Furthermore, for any unifier θ of S, we have $\theta = \sigma; \theta$.*

Proof. We simply observe that for any θ, $\theta(x_i) = \theta(t_i) = \theta(\sigma(x_i))$ for $1 \leq i \leq n$, and $\theta(x) = \theta(\sigma(x))$ otherwise. Clearly σ is an *mgu*, and since $D(\sigma) \cap I(\sigma) = \emptyset$ by the definition of solved forms, it is idempotent. $\square$

The next question is to find sets of transformations for solving unification problems. The following properties of such a set $\mathcal{T}$ of transformations are desirable:

1. (Soundness) Whenever $S \stackrel{*}{\Longrightarrow}_{\mathcal{T}} S'$, then $U(S') \subseteq U(S)$.
2. (Completeness) For any unifier θ of S, there is some solved S' s.t. $S \stackrel{*}{\Longrightarrow}_{\mathcal{T}} S'$, and $\sigma_{S'} \leq_E \theta[Var(S)]$.

When $\mathcal{T}$ satisfies (1) and (2), we say that $\mathcal{T}$ *is a complete set of transformations*. We also want the transformations to be as *deterministic as possible*, to reduce the search space. The set of transformations given in the next definition is a variant of the Herbrand–Martelli–Montanari transformations.

Definition 5.10 (The Set of Transformations $\mathcal{ST}$) Let S be any term system (possibly empty), and u, v two terms. The set $\mathcal{T}$ consists of the following transformations:

$$\{\langle u, u\rangle\} \cup S \Longrightarrow S \tag{triv}$$

$$\{\langle f(u_1, \ldots, u_k), f(v_1, \ldots, v_k)\rangle\} \cup S \Longrightarrow \{\langle u_1, v_1\rangle, \ldots, \langle u_k, v_k\rangle\} \cup S \tag{dec}$$

$$\{\langle x, v\rangle\} \cup S \Longrightarrow \{\langle x, v\rangle\} \cup S[v/x], \tag{vel}$$

where x is a variable s.t. $\langle x, v\rangle$ is **not** solved in $\{\langle x, v\rangle\} \cup S$, and $x \notin Var(v)$.

It should be noted that in transformation (*vel*), one should not relax the condition "$\langle x, v\rangle$ is **not** solved" to "x is not solved". If this were allowed, one could eliminate the variable x even when v is also a solved variable, and this could have the effect that v could then become unsolved again, leading to an infinite (cyclic) sequence of transformations.

The set $\mathcal{ST}$ is a complete set of transformations for standard unification. The basic idea of the transformations is to transform the original problem into a solved form which represents its own solution.

Example 5.11

$$\begin{aligned}
&\langle f(x, g(a, y)),\ f(x, g(y, x))\rangle \\
&\Longrightarrow_{dec}\ \langle x, x\rangle,\ \langle g(a, y), g(y, x)\rangle \\
&\Longrightarrow_{triv}\ \langle g(a, y), g(y, x)\rangle \\
&\Longrightarrow_{dec}\ \langle a, y\rangle,\ \langle y, x\rangle \\
&\Longrightarrow_{vel}\ \langle a, y\rangle,\ \langle a, x\rangle\,.
\end{aligned}$$

The reader can immediately verify that the substitution $[a/y, a/x]$ is a unifier of the original system (in fact, it is an *mgu*). The sense in which these transformations preserve the logically invariant properties of a unification problem is shown in the next lemma.

Lemma 5.12 *Let the set of all standard unifiers of a system S be denoted by $U(S)$. If $S \Longrightarrow S'$ using any transformation from $\mathcal{ST}$, then $U(S) = U(S')$.*

Proof. The only difficulty concerns (vel). Suppose $\{\langle x, v\rangle\} \cup S \Longrightarrow_{vel} \{\langle x, v\rangle\} \cup \sigma(S)$ with $\sigma = [v/x]$. For any substitution θ, if $\theta(x) = \theta(v)$, then $\theta = \sigma;\theta$, since $\sigma;\theta$ differs from θ only at x, but $\theta(x) = \theta(v) = \sigma;\theta(x)$. Thus,

$$\begin{aligned} & \theta \in U(\{\langle x, v\rangle\} \cup S) \\ \text{iff} \quad & \theta(x) = \theta(v) \ \text{ and } \ \theta \in U(S) \\ \text{iff} \quad & \theta(x) = \theta(v) \ \text{ and } \ \sigma;\theta \in U(S) \\ \text{iff} \quad & \theta(x) = \theta(v) \ \text{ and } \ \theta \in U(\sigma(S)) \\ \text{iff} \quad & \theta \in U(\{\langle x, v\rangle\} \cup \sigma(S)). \end{aligned}$$

□

The point here is that the most important feature of a unification problem—its set of solutions—is preserved under these transformations, and hence we are justified in our method of attempting to transform such problems into a trivial (solved) form in which the existence of an *mgu* is evident.

We may now show the soundness and completeness of these transformations following [49].

Theorem 5.13 (Soundness) *If $S \stackrel{*}{\Longrightarrow} S'$ with S' in solved form, then $\sigma_{S'} \in U(S)$.*

Proof. Using the previous lemma and a trivial induction on the length of transformation sequences, we see that $U(S) = U(S')$, and so clearly $\sigma_{S'} \in U(S)$. □

Theorem 5.14 (Completeness) *Every sequence of transformations*

$$S = S_0 \Longrightarrow S_1 \Longrightarrow S_2 \Longrightarrow \ldots$$

must eventually terminate. Furthermore, S is unifiable iff every system S' derivable from S is in solved form, and for every $\theta \in U(S)$, $\sigma_{S'} \leq \theta$.

Proof. We first show that every transformation sequence terminates. For any system S, let us define a complexity measure $\mu(S) = \langle n, m\rangle$, where n is the number of *unsolved* variables in the system, and m is the sum of the sizes of all the terms in the system. Then the lexicographic ordering on $\langle n, m\rangle$ is well-founded, and each transformation produces a new system with a measure strictly smaller under this ordering: $(triv)$ and (dec) must decrease m and can not increase n, and (vel) must decrease n.

Therefore the relation $\Longrightarrow$ is well-founded, and every transformation sequence must end in some system to which no transformation applies. Suppose a given sequence ends in a system S'. Now $\theta \in U(S)$ implies by lemma 5.12 that $\theta \in U(S')$, and so S' can contain no pairs of the form $\langle f(t_1, \ldots, t_n), g(t'_1, \ldots, t'_m)\rangle$ or of the form $\langle x, t\rangle$ with $x \in Var(t)$. But since no transformation applies, all pairs in S' must be in solved form. Finally, since $\theta \in U(S')$, by lemma 5.9 we must have $\sigma_{S'} \leq \theta$. $\square$

Putting these two theorems together, we have that the set $\mathcal{ST}$ can always find an mgu for a unifiable system of terms; as remarked in [49], this abstract formulation can be used to model many different unification algorithms, by simply specifying data structures and a control strategy.

The first published unification algorithm due to Robinson ([61]) can run in exponential time. Since Robinson's seminal discovery, several polynomial-time algorithms for standard unification have been given, including one by Robinson himself. Among them, we single out a quasi-linear algorithm due to Huet [35] (1976), a linear-time algorithm due to Paterson and Wegman [56] (1978), and quasi-linear and linear algorithms due to Martelli and Montanari [49] (1982). For an excellent account of standard unification, the reader is referred to Knight's survey article [43], and to Jouannaud and Kirchner's survey article [38]. Martelli and Montanari's important contribution ([49]), perhaps even more than their algorithm itself,[4] is to have demonstrated with perfect clarity that the method of transformations is remarkably well suited for tackling unification problems. In some sense, Martelli and Montanari revived Herbrand's approach, which led to new important work by Kirchner and others. In the next section, we sketch some versions of fast unification algorithms.

6 Fast (Standard) Unification

If one looks closely at the set of transformations $\mathcal{ST}$, one realizes that the complexity of unification algorithms is related to explicit variable elimination. Thus, a main concern in designing fast unification algorithms is to avoid explicit variable elimination. We first give the intuition behind a fast unification algorithm due to Martelli and Montanari [49].

Starting from a system S, suppose we only apply decomposition and deletion of trivial rules If no failure takes place and there are still pairs left, we must reach a system S' such that every pair is of the form $\langle x, v\rangle$, where x is a variable.

We can group all pairs sharing some common element to form equivalence classes.

[4] Their algorithm is not so different from Paterson and Wegman's algorithm.

Thus, S' can be viewed as a partition, where every class is of the form

$$\{x_1, \ldots, x_k, t_1, \ldots, t_m\},$$

where $x_1, \ldots, x_k$ are variables ($k > 0$) and $t_1, \ldots, t_m$ are nonvariable terms ($m \geq 0$).

Clearly, S' is unifiable only if for every class, all nonvariable terms have the same root symbol. The other basic idea is to **analyze dependencies among variables** (analogy with solving systems of linear equations).

A precedence relation on classes can be defined as follows:

$C < C'$ iff C' contains some variable x that occurs in some nonvariable term t in C.

Intuitively, the class of C *needs* the value of the variable x. The following is easily shown.

Lemma 6.1 *If $<^+$ is reflexive, then S' is not unifiable.*

From now on, we are dealing with sets of the form

$$\{x_1, \ldots, x_k, t_1, \ldots, t_m\},$$

that Martelli and Montanari call *multiequations*, and write in the form

$$\{x_1, \ldots, x_k\} := \{t_1, \ldots, t_m\}.$$

Since eliminating duplicate terms may be costly, they allow both sides to be multisets. If $<^+$ is acyclic (irreflexive), roughly speaking, Martelli and Montanari do the following:

Pick some class $\{x_1, \ldots, x_k, t_1, \ldots, t_m\}$ where $m > 0$ and the t_i are not constants. Since all the t_i have the same root symbol, say f, form the new system in which the above class is replaced by the sets

$$\{x_1, \ldots, x_k, f(y_1, \ldots, y_n)\}, \{y_1, t_1/1, \ldots, t_m/1\}, \ldots, \{y_n, t_1/n, \ldots, t_m/n\}.$$

Again, group blocks together to form a partition, and check for acyclicity of the new $<^+$. Continue this process until failure, or no new classes are formed. If the last $<^+$ obtained is acyclic, we can form a *mgu* in *triangular form*, by using any total ordering of the classes extending $<^+$. Formally, a triangular form is defined as follows.

Definition 6.2 Given an idempotent substitution σ (i.e., $D(\sigma) \cap I(\sigma) = \emptyset$) with domain $D(\sigma) = \{x_1, \ldots, x_k\}$, a *triangular form for* σ is a finite set T of pairs $\langle x, t\rangle$ where $x \in D(\sigma)$ and t is a term, such that this set T can be sorted (possibly in more than one way) into a sequence $\langle\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\rangle$ satisfying the following properties: for every i, $1 \le i \le k$,

(1) $\{x_1, \ldots, x_i\} \cap Var(t_i) = \emptyset$, and

(2) $\sigma = [t_1/x_1]\,; \ldots ;[t_k/x_k]$.

The set of variables $\{x_1, \ldots, x_k\}$ is called the *domain* of T. Note that in particular $x_i \notin Var(t_i)$ for every i, $1 \le i \le k$, but variables in the set $\{x_{i+1}, \ldots, x_k\}$ may occur in $t_1, \ldots, t_i$. It is easily seen that σ is an (idempotent) *mgu* of the term system T.

Example 6.3 Consider the substitution $\sigma = [f(f(x_3, x_3), f(x_3, x_3))/x_1, f(x_3, x_3)/x_2]$. The system $T = \{\langle x_1, f(x_2, x_2)\rangle, \langle x_2, f(x_3, x_3)\rangle\}$ is a triangular form of σ since it can be ordered as $\langle\langle x_1, f(x_2, x_2)\rangle, \langle x_2, f(x_3, x_3)\rangle\rangle$ and $\sigma = [f(x_2, x_2)/x_1]\,;[f(x_3, x_3)/x_2]$.

It turns out that we have computed a certain relation on terms, a *unification closure*. We now define this concept, due to Paterson and Wegman [56] (1978), and give a fast algorithm based on it.

Definition 6.4 Let Σ be a finite ranked alphabet (signature). Consider a finite graph G whose nodes are labeled with symbols in Σ or variables in $\mathcal{X}$. Let $\Lambda : V \to \Sigma \cup \mathcal{X}$ be the labeling function.

If $\Lambda(u)$ is a constant or a variable, then u is a terminal node; If $\Lambda(u)$ is a function symbol of rank k, then u has k immediate successors $u[1], \ldots, u[k]$.

An equivalence relation R on a graph G is a *unification closure* iff, for every pair (u, v) of nodes in V^2, whenever uRv then:

(1) Either $\Lambda(u) = \Lambda(v)$, or one of $\Lambda(u)$, $\Lambda(v)$ is a variable;

(2) If $\Lambda(u) = \Lambda(v)$ and $r(\Lambda(u)) = n$, then for every i, $1 \le i \le n$, $u[i]Rv[i]$.

Graphically, if u and v are two nodes labeled with the same symbol f of rank n, if $u[1], \ldots, u[n]$ are the successors of u and $v[1], \ldots, v[n]$ are the successors of v,

$$\begin{array}{ccccccc} & u\,. & & & & v\,. & \\ & & & & & & \\ u[1]\,. & \ldots & .\,u[n] & & v[1]\,. & \ldots & .\,v[n] \end{array}$$

if u and v are equivalent then $u[i]$ and $v[i]$ are equivalent for all $i, 1 \le i \le n$. We have a kind of forward closure. The following lemma is easily shown.

Lemma 6.5 *There is an algorithm which, given any arbitrary relation R_0 on a finite graph G, decides whether the smallest unification closure containing a relation R_0 on G exists, and if so computes it.*

In order to test whether a system $S = \{\langle u_1, v_1\rangle, \ldots, \langle u_m, v_m\rangle\}$ is unifiable, we can compute the unification closure of the relation S on the graph G_S constructed as follows:

(i) The set set of nodes of G_S is the set of all subterms of terms in S.

(ii) Every subterm that is either a constant or a variable is a terminal node labeled with that symbol.

(iii) For every subterm of the form $fs_1 \ldots s_k$, the label is f, and there is an edge from $fs_1 \ldots s_k$ to s_i, for each i, $1 \leq i \leq k$.

Let R be the least unification closure containing the relation S on the graph G_S, if it exits. A new graph G_S/R can be constructed as follows:

(i) The nodes of G_S/R are the equivalence classes of R.

(ii) There is an edge from a class C to a class C' iff there is an edge in G_S from some node s in class C to some node t in class C'.

The following lemma is essentially due to Paterson and Wegman [56] (1978).

Lemma 6.6 *The system S is unifiable iff the unification closure R exists and the graph G_S/R is acyclic.*

When S is unifiable, let $\langle C_1, \ldots, C_n\rangle$ be the sequence of all equivalence classes containing some variable, ordered such that, if there is a path from C_i to C_j, then $i < j$.

For each i, $1 \leq i \leq n$, if C_i contains some nonvariable term, let t_i be any such term, else let t_i be any variable in C_i.

Let

$$\sigma_i = [t_i/z_1, \ldots, t_i/z_k],$$

where $\{z_1, \ldots, z_k\}$ is the set of variables in C_i, and let

$$\sigma = \sigma_1 ; \ldots ; \sigma_n.$$

Then σ is a most general unifier of S. The substitution σ has a triangular representation.

Example 6.7 Consider the pair

$$\langle f(f(x_2, x_2), f(x_3, x_3)), f(x_1, x_2)\rangle.$$

The nontrivial classes of the unification closure containing variables are

$$\{x_1, f(x_2, x_2)\}, \{x_2, f(x_3, x_3)\}.$$

The first class precedes the second. A triangular form of the *mgu* σ is

$$\sigma = [f(x_2, x_2)/x_1]\,; [f(x_3, x_3)/x_2].$$

Note that

$$\sigma = [f(f(x_3, x_3), f(x_3, x_3))/x_1, f(x_3, x_3)/x_2].$$

We now give a simple fast algorithm, assuming for simplicity that every symbol in Σ is either a constant or binary. This algorithm is basically Ravi Sethi's algorithm, in [1].

```
procedure unif(u, v : node; var R : partition; var flag : bool);
  var s, t : node; flag1, flag2 : bool;
  begin
    flag1 := false; flag2 := false;
    s := find(R, u); t := find(R, v);
    if s = t then
      flag := true
    else
      if nonvar(u) and nonvar(v) and root(u) = root(v) then
        union(R, s, t);
        unif(left(u), left(v), R, flag1);
        unif(right(u), right(v), R, flag2);
        flag := flag1 and flag2
      else
        if var(u) or var(v) then
          union(R, s, t);
          flag := true
        else
          flag := false
        endif
      endif
```

endif;
if $flag$ **and** $acyclic(R)$ **then**
$printsubst(R)$
else
$failure(R)$
endif
end

Using Tarjan's fast version of *union* and $find$, the algorithm runs in time $O(n\alpha(n))$, where $\alpha(n)$ is a sort of inverse of Ackermann's function that grows extremely slowly.

We conclude this section with some comments on Paterson and Wegman's fast algorithm [56] (1978). Paterson and Wegman's algorithm is a unification closure algorithm, and it uses the concept of a "root class". A *root class* is an equivalence class of nodes containing only terms corresponding to nodes of the DAG that have no parents. It is immediately seen that if a system S is unifiable, then every class that is mimimal in the ordering defined such that $C < C'$ iff there is a path from C to C' in the DAG, is a root class.

Paterson and Wegman's algorithm processes root classes first. A root class has the property that no further nodes can be added to it as the result of computing a unification closure, because propagation proceeds from parent to children. Thus, once a root class has been processed, it can be deleted.

It should be noted that Paterson and Wegman's original algorithm contains bugs. The bugs were reported and fixed by De Champeaux [19]. One of the bugs is a trivial typo. The other bug is more subtle. In the Paterson-Wegman's algorithm, the equivalence of two elements is represented by the existence of a special (undirected) edge between these two elements (not to be confused with the edges of the DAG). The problem is that multiple edges can be created by the algorithm, but this is not taken into account by the algorithm.

We now come back to the method of matings in the general case of languages with equality.

7 Equational Matings

In this section, we show that the method of matings for languages **without** equality presented in section 4 can be generalized to languages **with** equality. This generalization will lead us to a decidable form of unification extending standard unification and called *rigid E-unification*. The generalized method of matings was first presented in Gallier, Raatz, and Snyder [23] (1987), where it was conjectured that rigid E-unification is decidable. Several months later, Gallier, Narendran, Plaisted, and Snyder proved that rigid E-unification is NP-complete and that finite complete sets of rigid E-unifiers always exist. These results were announced (without complete proofs) at LICS'88 [25]. Full details and proofs appear in Gallier, Narendran, Plaisted, and Snyder [27]. A detailed presentation of the method of equational matings is given in [23, 26], or [28].

First, it is important to note that lemma 3.8, theorem 3.11, theorem 4.3, and lemma 4.5, also hold for languages **with** equality. The main difference with the case of languages without equality, is that the criterion for checking whether a vertical path is unsatisfiable is more complicated, and involves some equality reasoning.

A criterion for the unsatisfiability of a conjunction of literals based on the concept of congruence closure is known. In order to explain this criterion, it is convenient to represent every atomic formula as an equation. This can be done by adding to our language (which already contains the special sort *bool*) the constant $\top$ of sort *bool*, interpreted as **true**. Then, every atomic formula $Pt_1 \ldots t_n$ of sort *bool* can be expressed as the equation $(Pt_1 \ldots t_n \doteq \top)$. Hence, we can assume that all atomic formulae are equations. The notations $Pt_1 \ldots t_n$ and $(Pt_1 \ldots t_n \doteq \top)$ will be used interchangeably for atomic formulae of sort *bool*.

Given a vertical path π, we can arrange the literals in π by grouping positive and negative literals together, to form a conjunction C_π of the form

$$(s_1 \doteq t_1) \wedge \ldots \wedge (s_m \doteq t_m) \wedge \neg(s'_1 \doteq t'_1) \wedge \ldots \wedge \neg(s'_n \doteq t'_n).$$

The congruence closure method defined below enables us to decide whether conjunctions of the above form are satisfiable or not.

Definition 7.1 (Congruence closure) Let $TERMS(\pi)$ be the set of all subterms of terms in π. Construct the labeled directed graph G_π as follows:

- The set of Nodes of G_π is $TERMS(\pi)$.
- The node $f(t_1, \ldots, t_n)$ is labeled with f.
- For each node $f(t_1, \ldots, t_n)$, there is an edge from $f(t_1, \ldots, t_n)$ to each t_i.

A relation $\simeq$ on the set of nodes of G_π is *G-congruential* iff, for any two nodes $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$, if $s_i \simeq t_i, 1 \leq i \leq n$, then $f(s_1, \ldots, s_n) \simeq f(t_1, \ldots, t_n)$.

Given a vertical path

$$\pi = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s'_1 \doteq t'_1), \ldots, \neg(s'_n \doteq t'_n)\},$$

let $E = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m)\}$. The following results can be shown (see Kozen [45], Nelson and Oppen [55], or Gallier [22]).

Lemma 7.2 *There is a smallest G-congruential equivalence relation on G_π containing E. It is called the congruence closure of E, and it is denoted as $\overset{*}{\cong}_E$.*

Lemma 7.3 *A vertical path π is unsatisfiable iff $s'_j \overset{*}{\cong}_E t'_j$ for some j, $1 \leq j \leq n$.*

The congruence closure $\overset{*}{\cong}_E$ can be computed in polynomial time (Kozen [45,46], Nelson and Oppen [55], Downey, Sethi, and Tarjan [21]).

Now, recall that theorem 4.3 states that a universal sentence A in *nnf* is unsatisfiable iff there is some amplification D of A and some (ground) substitution σ such that $\sigma(D)$ is unsatisfiable. Since $\sigma(D)$ is quantifier-free, by lemma 4.5, $\sigma(D)$ is unsatifiable iff all vertical paths in $\sigma(D)$ are unsatisfiable. In view of lemma 7.3, we can now give the criterion stating that vertical paths in $\sigma(D)$ are unsatisfiable:

> Given any $\pi = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s'_1 \doteq t'_1), \ldots, \neg(s'_n \doteq t'_n)\}$ in $vp(D)$,[5] there is some i $(1 \leq i \leq n)$, such that $\{\sigma(s_1 \doteq t_1), \ldots, \sigma(s_m \doteq t_m), \neg\sigma(s'_i \doteq t'_i)\}$ is unsatisfiable.

Since the set $\{\sigma(s_1 \doteq t_1), \ldots, \sigma(s_m \doteq t_m), \neg\sigma(s'_i \doteq t'_i)\}$ consists of quantifier-free formulae, it is unsatisfiable iff $\sigma(s'_i \doteq t'_i)$ is provable from $\{\sigma(s_1 \doteq t_1), \ldots, \sigma(s_m \doteq t_m)\}$, treated as a set of *ground equations*. By lemma 7.3, this is equivalent to saying that

> $\sigma(s'_i)$ and $\sigma(t'_i)$ are congruent modulo the congruence closure associated with the set of equations $\{\sigma(s_1 \doteq t_1), \ldots, \sigma(s_m \doteq t_m)\}$.

The definition of an equational mating is motivated by the above observation. It is designed so that we have a criterion expressed in terms of vertical paths for testing whether given a quantifier-free formula D, there is some substitution σ such that $\sigma(D)$ is unsatisfiable (see lemma 7.5).

Definition 7.4 Let A be a quantifier-free formula in nnf. An *equational mating $\mathcal{M}$ for A* is a pair $\langle MS, \sigma\rangle$, where MS is a set of sets of literals called *mated sets* and σ is a substitution, such that, each mated set is a subset of some vertical path $\pi \in vp(A)$ and is of the form

$$\{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s \doteq t)\} \subseteq \pi,$$

where $m \geq 0$,[6] and, for every mated set $\{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s \doteq t)\} \in MS$, the set of literals $\{\sigma(s_1 \doteq t_1), \ldots, \sigma(s_m \doteq t_m), \neg\sigma(s \doteq t)\}$ is unsatisfiable. The substitution associated with the mating $\mathcal{M}$ is also denoted as $\sigma_{\mathcal{M}}$. We also commit a slight abuse of language (and notation) and say that a mated set belongs to $\mathcal{M}$.

---

[5] **Warning**: $\pi \in vp(D)$, **not** $\pi \in vp(\sigma(D))$.

[6] The case $m = 0$ is indeed possible when $\sigma(s) = \sigma(t)$, i.e., when σ is a unifier of s and t.

An equational mating $\mathcal{M}$ is a *refutation mating* iff $\sigma_{\mathcal{M}}(A)$ is unsatisfiable.

An equational mating $\mathcal{M}$ is *path acceptable*[7] (for short, *p-acceptable*), iff, for every path $\pi \in vp(A)$, there is some mated set $\{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s \doteq t)\} \in \mathcal{M}$, such that

$$\{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s \doteq t)\} \subseteq \pi.$$

A number of remarks are in order:

(1) Given the substitution σ, the mating condition can be tested using the congruence closure method. The difficulty is to decide whether or not the substitution σ exists.

(2) Given a family MS of mated sets, let $\vec{E} = (E_{\mathcal{S}})_{\mathcal{S} \in MS}$ be the family of sets of equations of the form $E_{\mathcal{S}} = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m)\}$ and $S = \{\langle s, t\rangle \mid \mathcal{S} \in MS\}$ the set of pairs where $E_{\mathcal{S}}$ and $\langle s, t\rangle$ are associated with the mated sets $\mathcal{S} = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m), \neg(s \doteq t)\} \in MS$. Observe that $\mathcal{M} = \langle MS, \sigma\rangle$ is a mating iff σ is a solution of the following problem:

Problem 1: Given $\vec{E} = \{E_i \mid 1 \leq i \leq n\}$ a family of n finite sets of equations and $S = \{\langle u_i, v_i\rangle \mid 1 \leq i \leq n\}$ a set of n pairs of terms, is there a substitution θ such that, treating each set $\theta(E_i)$ as a set of *ground* equations (i.e. holding the variables in $\theta(E_i)$ "rigid"), $\theta(u_i)$ and $\theta(v_i)$ are provably equal from $\theta(E_i)$ for $i = 1, \ldots, n$?

Equivalently, is there a substitution θ such that $\theta(u_i)$ and $\theta(v_i)$ can be shown congruent from $\theta(E_i)$ by the congruence closure method for $i = 1, \ldots, n$?

Problem 1 is a unification problem more general than standard unification. A substitution θ solving the above problem is called a *rigid $\vec{E}$-unifier of S*, and a pair $\langle \vec{E}, S\rangle$ such that S has some rigid $\vec{E}$-unifier is called an *equational premating*. This key observation is used in searching for the substitutions associated with matings. They are the rigid $\vec{E}$-unifiers of S.

The following lemma is a straightforward generalization of a lemma 4.7 to languages with equality.

Lemma 7.5 *Given a quantifier-free formula A in nnf, the following properties hold:*

(1) Given a substitution θ, if $\theta(A)$ is unsatisfiable, then there is a p-acceptable equational mating $\mathcal{M}$ for A.

(2) A p-acceptable equational mating $\mathcal{M}$ for A is a refutation mating for A, i.e. $\sigma_{\mathcal{M}}(A)$ is unsatisfiable.

---

[7] A path acceptable mating is also called a *spanning mating* by Miller [52].

Corollary 7.6 *Given a quantifier-free formula A in nnf, there is a substitution θ such that $\theta(A)$ is unsatisfiable iff there is a p-acceptable equational mating $\mathcal{M}$ for A.* □

As in section 4, the completeness and soundness for the method of equational matings is an immediate consequence of theorem 4.3, lemma 4.5, and lemma 7.5.

Theorem 7.7 *Given a universal sentence A in nnf, A is unsatisfiable iff some amplification D of A has a p-acceptable equational mating.*

Let us give some examples illustrating the use of theorem 7.7. From this point on, for the sake of brevity, we will use interchangeably the terms equational mating and mating.

Example 7.8 Consider the following Horn formula A, where x, y, z denote variables:

$$\begin{aligned}
&(a \doteq b) \wedge \\
&((f^3x \doteq x) \vee \neg(fx \doteq fb)) \wedge \\
&(Qa \vee \neg(f^3a \doteq a)) \wedge \\
&((f^5y \doteq y) \vee \neg Qy) \wedge \\
&(Ra \vee \neg(fa \doteq a) \vee \neg Pfa) \wedge \\
&\neg Rfz \wedge \\
&Pa
\end{aligned}$$

There are 24 vertical paths in A. Let $\theta = [a/x, a/y, a/z]$. The substitution θ closes all the paths in $\theta(A)$, which is easy to see for the 21 vertical paths containing the sets of literals $\{(f^3a \doteq a), \neg(f^3a \doteq a)\}$, $\{Qa, \neg Qa\}$, and $\{(a \doteq b), \neg(fa \doteq fb)\}$. A p-acceptable mating for A is given by θ and the following set of 6 sets of literals:

$$\begin{gathered}
\{\{(f^3x \doteq x), \neg(f^3a \doteq a)\}, \\
\{Qa, \neg Qy\}, \\
\{(a \doteq b), \neg(fx \doteq fb)\}, \\
\{(f^5y \doteq y), (f^3x \doteq x), Ra, \neg Rfz\}, \\
\{(f^5y \doteq y), (f^3x \doteq x), \neg(fa \doteq a)\}, \\
\{(f^5y \doteq y), (f^3x \doteq x), Pa, \neg Pfa\}\}.
\end{gathered}$$

The above set is a mating because $(fa \doteq a)$ is equationally provable from $(f^3a \doteq a)$ and $(f^5a \doteq a)$. Indeed, $(f^3a \doteq a)$ implies $(f^4a \doteq fa)$, which implies $(f^5a \doteq f^2a)$, which, by transitivity, implies $(f^2a \doteq a)$. In turn, $(f^2a \doteq a)$ implies $(f^3a \doteq fa)$, and by one more application of transitivity, this implies $(fa \doteq a)$. According to lemma 7.5, $\theta(A)$ is unsatisfiable.

Example 7.9 Let A be the following (equational) sentence:

$$\forall x\forall y\forall z(*(x,*(y,z)) \doteq *(*(x,y),z))) \wedge \tag{1}$$
$$\forall u(*(u,1) \doteq u) \wedge \tag{2}$$
$$\forall v(*(1,v) \doteq v) \wedge \tag{3}$$
$$\forall w(*(w,w) \doteq 1) \wedge \tag{4}$$
$$\neg(*(a,b) \doteq *(b,a)). \tag{5}$$

The first three equations are the axioms for monoids (a binary operation $*$ which is associative and has an identity element 1), the fourth equation asserts that the square of every element is the identity, and the fifth asserts the negation of the commutativity of $*$ (A is the result of a Skolemization). The unsatisfiability of A asserts that any monoid such that the square of every element is the identity is commutative.

Consider the following amplification D of A in the left column and the set MS consisting of one set of literals in the right column:

$$\begin{aligned}
D = {} & (*(u_1,1) \doteq u_1) \\
& \wedge (*(w_1,w_1) \doteq 1) \\
& \wedge (*(x_1,*(y_1,z_1)) \doteq *(*(x_1,y_1),z_1))) \\
& \wedge (*(x_2,*(y_2,z_2)) \doteq *(*(x_2,y_2),z_2))) \\
& \wedge (*(w_2,w_2) \doteq 1) \\
& \wedge (*(1,v_1) \doteq v_1) \\
& \wedge (*(x_3,*(y_3,z_3)) \doteq *(*(x_3,y_3),z_3))) \\
& \wedge (*(x_4,*(y_4,z_4)) \doteq *(*(x_4,y_4),z_4))) \\
& \wedge (*(w_3,w_3) \doteq 1) \\
& \wedge \neg(*(a,b) \doteq *(b,a)).
\end{aligned}
\qquad
\begin{aligned}
MS = \{\{ & (*(u_1,1) \doteq u_1), \\
& (*(w_1,w_1) \doteq 1), \\
& (*(x_1,*(y_1,z_1)) \doteq *(*(x_1,y_1),z_1))), \\
& (*(x_2,*(y_2,z_2)) \doteq *(*(x_2,y_2),z_2))), \\
& (*(w_2,w_2) \doteq 1), \\
& (*(1,v_1) \doteq v_1), \\
& (*(x_3,*(y_3,z_3)) \doteq *(*(x_3,y_3),z_3))), \\
& (*(x_4,*(y_4,z_4)) \doteq *(*(x_4,y_4),z_4))), \\
& (*(w_3,w_3) \doteq 1), \\
& \neg(*(a,b) \doteq *(b,a))\}\}.
\end{aligned}$$

Let θ be the substitution

$$\begin{aligned}
&[a/u_1, (a*b)/w_1, a/x_1, (a*b)/y_1, (a*b)/z_1, \\
&\ a/x_2, a/y_2, b/z_2, a/w_2, b/v_1, \\
&\ b/x_3, (a*b)/y_3, b/z_3, a/x_4, b/y_4, b/z_4, b/w_3].
\end{aligned}$$

We claim that $\langle MS, \theta\rangle$ is a mating for D. For simplicity of notation let us adopt infix notation, and denote $*(s,t)$ as $s*t$. Then, we have:

$$\begin{aligned}
a*b &= \{a*1\}*b && \text{by (2)} \\
&= \{a*[(a*b)*(a*b)]\}*b && \text{by (4)}
\end{aligned}$$

$$
\begin{aligned}
&= \{[a*(a*b)]*(a*b)\}*b && \text{by } (1)\\
&= \{[(a*a)*b]*(a*b)\}*b && \text{by } (1)\\
&= \{[1*b]*(a*b)\}*b && \text{by } (4)\\
&= \{b*(a*b)\}*b && \text{by } (3)\\
&= b*\{(a*b)*b\} && \text{by } (1)\\
&= b*\{a*(b*b)\} && \text{by } (1)\\
&= b*\{a*1\} && \text{by } (4)\\
&= b*a, && \text{by } (2)
\end{aligned}
$$

which shows that $\langle MS, \theta\rangle$ is a p-acceptable equational mating for D (there is a single vertical path in D).

8 Rigid E-Unification

In the second remark following definition 7.4, it was noted that a new form of unification, "rigid E-unification", arises naturally in extending Andrews and Bibel's theorem proving method of matings [4, 6, 11, 12, 13], to first-order languages with equality. What was noted in this remark, is that $\mathcal{M} = \langle MS, \sigma\rangle$ is a mating iff σ is a solution of the following problem:

Problem 1: Given $\vec{E} = \{E_i \mid 1 \leq i \leq n\}$ a family of n finite sets of equations and $S = \{\langle u_i, v_i\rangle \mid 1 \leq i \leq n\}$ a set of n pairs of terms, is there a substitution θ such that, treating each set $\theta(E_i)$ as a set of *ground* equations (i.e. holding the variables in $\theta(E_i)$ "rigid"), $\theta(u_i)$ and $\theta(v_i)$ are provably equal from $\theta(E_i)$ for $i = 1, \ldots, n$?

Equivalently, is there a substitution θ such that $\theta(u_i)$ and $\theta(v_i)$ can be shown congruent from $\theta(E_i)$ by the congruence closure method for $i = 1, \ldots, n$?

Actually, it turns out that problem 1 reduces to the following simpler problem:

Problem 2: Given a finite set $E = \{u_1 \doteq v_1, \ldots, u_n \doteq v_n\}$ of equations and a pair $\langle u, v\rangle$ of terms, is there a substitution θ such that, treating $\theta(E)$ as a set of ground equations, $\theta(u) \overset{*}{\longleftrightarrow}_{\theta(E)} \theta(v)$, that is, $\theta(u)$ and $\theta(v)$ are congruent modulo $\theta(E)$ by congruence closure (Kozen [45], Nelson and Oppen [55])?

The substitution θ is called a *rigid E-unifier of u and v.*

Example 8.1 Let $E = \{fa \doteq a,\ ggx \doteq fa\}$, and $\langle u, v\rangle = \langle gggx, x\rangle$. Then, the substitution $\theta = [ga/x]$ is a rigid E-unifier of u and v. Indeed, $\theta(E) = \{fa \doteq a,\ ggga \doteq fa\}$, and

$\theta(gggx)$ and $\theta(x)$ are congruent modulo $\theta(E)$, since

$$\begin{aligned} \theta(gggx) = gggga &\longrightarrow gfa && \text{using } ggga \doteq fa \\ &\longrightarrow ga = \theta(x) && \text{using } fa \doteq a. \end{aligned}$$

Note that θ is not the only rigid E-unifier of u and v. For example, $[gfa/x]$ or more generally $[gf^n a/x]$ is a rigid E-unifier of u and v. However, θ is more general than all of these rigid E-unifiers (in a sense to be made precise later).

The importance of rigid E-unification stems from the fact that it is decidable, and in fact NP-complete, see Gallier, Narendran, Plaisted, and Snyder [27]. Remarkably, it can also be shown that there is always a finite set of most general rigid E-unifiers called a complete set of rigid E-unifiers [27].

It is interesting to observe that the notion of rigid E-unification arises by *bounding* the resources, in this case, the number of available instances of equations in E. In order to understand more clearly the concept of rigid E-unification, let us recall what (unrestricted) E-unification is. We are given a set of equations $E = \{u_1 \doteq v_1, \ldots, u_n \doteq v_n\}$, and (for simplicity) a pair of terms $\langle u, v\rangle$. The problem is to decide whether is there a substitution θ s.t. $\theta(u) \overset{*}{\longleftrightarrow}_E \theta(v)$.

Note that there is *no bound* on the number of instances of equations in E that can be used in the proof that $\theta(u) \overset{*}{\longleftrightarrow}_E \theta(v)$. Going back to definition 2.16, we observe that $\theta(u) \overset{*}{\longleftrightarrow}_E \theta(v)$ iff is there a *multiset* of equations (from E)

$$\left\{ \binom{u'_1 \doteq v'_1}{n_1}, \ldots, \binom{u'_m \doteq v'_m}{n_m} \right\}$$

and m sets of substitutions $\{\sigma_{j,1}, \ldots, \sigma_{j,n_j}\}$, s.t., letting

$$E' = \{\sigma_{j,k}(u'_j \doteq v'_j) \mid 1 \le j \le m,\ 1 \le k \le n_j\},$$

we have $\theta(u) \overset{*}{\longleftrightarrow}_{E'} \theta(v)$, considering E' as **ground**. Basically, the restriction imposed by rigid E-unification is that $n_1 = \ldots = n_m = 1$, i.e., at most a single instance of each equation in E can be used. In fact, these instances $\theta(u_1 \doteq v_1), \ldots, \theta(u_n \doteq v_n)$ must arise from the substitution θ itself. Also, once these instances have been created, the remaining variables (if any) are considered rigid, that is, treated as constants, so that it is not possible to instantiate these instances. Thus, rigid E-unification and Girard's linear logic [29] share the same spirit. Since the resources are bounded, it is not too surprising that rigid E-unification is decidable, but it is not obvious at all that the problem is in NP. The special case of rigid E-unification where E is a set of ground equations has been investigated by

Kozen who has shown that this problem is NP-complete (Kozen, [45,46]). Thus, rigid E-unification is NP-hard. We also showed that it is in NP, hence NP-complete.

Our plan for the rest of this section is to define precisely what complete sets of rigid E-unifiers are, and to sketch the decision procedure. The definitions of a rigid E-unifier, the preorder $\leq_E$, and complete sets of rigid E-unifiers, will parallel those given for E-unification, but equations are considered as ground in equational proofs. It will be convenient to write $u \overset{*}{\cong}_E v$ to express that $u \overset{*}{\longleftrightarrow}_E v$, treating the equations in E as *ground equations*.

Definition 8.2 Let $E = \{(s_1 \doteq t_1), \ldots, (s_m \doteq t_m)\}$ be a finite set of equations, and let $Var(E) = \bigcup_{(s \doteq t) \in E} Var(s \doteq t)$ denote the set of variables occurring in E.[8] Given a substitution θ, we let $\theta(E) = \{\theta(s_i \doteq t_i) \mid s_i \doteq t_i \in E,\ \theta(s_i) \neq \theta(t_i)\}$. Given any two terms u and v,[9] a substitution θ is a *rigid unifier of u and v modulo E* (for short, a *rigid E-unifier of u and v*) iff

> $\theta(u) \overset{*}{\cong}_{\theta(E)} \theta(v)$, that is, $\theta(u)$ and $\theta(v)$ are congruent modulo the set $\theta(E)$ considered as a set of *ground* equations.

The following example should help grasping the notion of rigid E-unification. The problem is to show that if $x \cdot x = 1$ in a monoid, then the monoid is commutative.

Example 8.3

$$
\begin{aligned}
E = \{u_1 \cdot 1 &\doteq u_1 \\
w_1 \cdot w_1 &\doteq 1 \\
x_1 \cdot (y_1 \cdot z_1) &\doteq (x_1 \cdot y_1) \cdot z_1 \\
x_2 \cdot (y_2 \cdot z_2) &\doteq (x_2 \cdot y_2) \cdot z_2 \\
w_2 \cdot w_2 &\doteq 1 \\
1 \cdot v_1 &\doteq v_1 \\
x_3 \cdot (y_3 \cdot z_3) &\doteq (x_3 \cdot y_3) \cdot z_3 \\
x_4 \cdot (y_4 \cdot z_4) &\doteq (x_4 \cdot y_4) \cdot z_4 \\
w_3 \cdot w_3 &\doteq 1\}.
\end{aligned}
$$

$$\langle u, v \rangle = \langle a \cdot b,\ b \cdot a \rangle.$$

[8] It is possible that equations have variables in common.

[9] It is possible that u and v have variables in common with the equations in E.

The reader can verify that θ below is a rigid E-unifier:

$$\begin{aligned}\theta = [&a/u_1,\ a/x_1,\ a/x_2,\ a/y_2,\ a/w_2,\ a/x_4,\\ &b/z_2,\ b/v_1,\ b/x_3,\ b/z_3,\ b/y_4,\ b/z_4,\ b/w_3,\\ &a\cdot b/w_1,\ a\cdot b/y_1,\ a\cdot b/z_1,\ a\cdot b/y_3].\end{aligned}$$

Definition 8.4 Let E be a (finite) set of equations, and W a (finite) set of variables. For any two substitutions σ and θ, $\sigma =_E \theta[W]$ iff $\sigma(x) \overset{*}{\cong}_E \theta(x)$ for every $x \in W$. The relation $\sqsubseteq_E$ is defined as follows. For any two substitutions σ and θ, $\sigma \sqsubseteq_E \theta[W]$ iff $\sigma =_{\theta(E)} \theta[W]$. The set W is omitted when $W = \mathcal{X}$ (where $\mathcal{X}$ is the set of variables), and similarly E is omitted when $E = \emptyset$.

Intuitively speaking, $\sigma \sqsubseteq_E \theta$ iff σ can be generated from θ using the equations in $\theta(E)$. Clearly, $\sqsubseteq_E$ is reflexive. However, it is not symmetric as shown by the following example.

Example 8.5 Let $E = \{fx \doteq x\}$, $\sigma = [fa/x]$ and $\theta = [a/x]$. Then $\theta(E) = \{fa \doteq a\}$ and $\sigma(x) = fa \overset{*}{\cong}_{\theta(E)} a = \theta(x)$, and so $\sigma \sqsubseteq_E \theta$. On the other hand $\sigma(E) = \{ffa \doteq fa\}$, but a and fa are not congruent from $\{ffa \doteq fa\}$. Thus $\theta \sqsubseteq_E \sigma$ *does not* hold.

It is not difficult to show that $\sqsubseteq_E$ is also transitive. We also need an extension of $\sqsubseteq_E$ defined as follows.

Definition 8.6 Let E be a (finite) set of equations, and W a (finite) set of variables. The relation $\leq_E$ is defined as follows: for any two substitutions σ and θ, $\sigma \leq_E \theta[W]$ iff $\sigma\,;\eta \sqsubseteq_E \theta[W]$ for some substitution η (that is, $\sigma\,;\eta =_{\theta(E)} \theta[W]$ for some η).

Intuitively speaking, $\sigma \leq_E \theta$ iff σ is more general than some substitution that can be generated from θ using $\theta(E)$. Clearly, $\leq_E$ is reflexive. The transitivity of $\leq_E$ is also shown easily. When $\sigma \leq_E \theta[W]$, we say that σ *is (rigid) more general than* θ *over* W. It can be shown that if σ is a rigid E-unifier of u and v and $\sigma \leq_E \theta$, then θ is a rigid E-unifier of u and v. The converse is false. Finally, the crucial concept of a complete set of rigid E-unifiers can be defined.

Definition 8.7 Given a (finite) set E of equations, for any two terms u and v, letting $V = Var(u) \cup Var(v) \cup Var(E)$, a set U of substitutions is a *complete set of rigid E-unifiers for u and v* iff: For every $\sigma \in U$,

(i) $D(\sigma) \subseteq V$ and $D(\sigma) \cap I(\sigma) = \emptyset$ (idempotence),

(ii) σ is a rigid E-unifier of u and v,

(iii) For every rigid E-unifier θ of u and v, there is some $\sigma \in U$, such that, $\sigma \leq_E \theta[V]$.

Suppose we want to find a rigid E-unifier θ of u and v. There is an algorithm using transformations for finding rigid E-unifiers. Roughly, the idea is to use a form of unfailing completion procedure (Knuth and Bendix [44], Huet [36], Bachmair [8], Bachmair, Dershowitz, and Plaisted [9], Bachmair, Dershowitz, and Hsiang [10]). In order to clarify the differences between our method and unfailing completion, especially for readers unfamiliar with this method, we briefly describe the use of unfailing completion as a refutation procedure. For more details, the reader is referred to Bachmair [8].

Let E be a set of equations, and $\succ$ a reduction ordering total on ground terms. The central concept is that of E being *ground Church-Rosser w.r.t.* $\succ$. The crucial observation is that every ground instance $\sigma(l) \doteq \sigma(r)$ of an equation $l \doteq r \in E$ is orientable w.r.t. $\succ$, since $\succ$ is total on ground terms. Let $E^{\succ}$ be the set of all instances $\sigma(l) \doteq \sigma(r)$ of equations $l \doteq r \in E \cup E^{-1}$ with $\sigma(l) \succ \sigma(r)$ (the set of *orientable instances*). We say that E is *ground Church-Rosser w.r.t.* $\succ$ iff for every two ground terms u, v, if $u \stackrel{*}{\longleftrightarrow}_E v$, then there is some ground term w such that $u \stackrel{*}{\longrightarrow}_{E^{\succ}} w$ and $w \stackrel{*}{\longleftarrow}_{E^{\succ}} v$. Such a proof is called a *rewrite proof*.

An unfailing completion procedure attempts to produce a set E^{∞} equivalent to E and such that E^{∞} is ground Church-Rosser w.r.t. $\succ$. In other words, every ground equation provable from E has a rewrite proof in E^{∞}. The main mechanism involved is the computation of critical pairs. Given two equations $l_1 \doteq r_1$ and $l_2 \doteq r_2$ where l_2 is unifiable with a subterm l_1/β of l_1 which is not a variable, the pair $\langle \sigma(l_1[\beta \leftarrow r_2]), \sigma(r_1) \rangle$ where σ is a *mgu* of l_1/β and l_2 is a *critical pair*.

If we wish to use an unfailing completion procedure as a refutation procedure, we add two new constants T and F and a new binary function symbol eq to our language. In order to prove that $E \models u \doteq v$ for a ground equation $u \doteq v$, we apply the unfailing completion procedure to the set $E \cup \{eq(u,v) \doteq F,\ eq(z,z) \doteq T\}$, where z is a new variable. It can be shown that $E \models u \doteq v$ iff the unfailing completion procedure generates the equation $F \doteq T$. Basically, given any proof of $F \doteq T$, the unfailing completion procedure extends E until a rewrite proof is obtained. It can be shown that unfailing completion is a complete refutation procedure, but of course, it is not a decision procedure. It should also be noted that when unfailing completion is used as a refutation procedure, E^{∞} is actually never generated. It is generated "by need", until $F \doteq T$ turns up.

We now come back to our situation. Without loss of generality, it can be assumed that we have a rigid E-unifier θ of T and F such that $\theta(E)$ is ground. In this case, equations in $\theta(E)$ are orientable instances. The crucial new idea is that in trying to obtain a rewrite proof of $F \doteq T$, we still compute critical pairs, but we **never rename variables**. If l_2 is equal to l_1/β, then we get a critical pair essentially by simplification. Otherwise, some variable in l_1 or in l_2 gets bound to a term *not* containing this variable. Thus the total number of variables in E keeps decreasing. Therefore, after a polynomial number of steps (in fact, the number of variables in E) we must stop or fail. So we get membership in NP. Oversimplifying a bit, we can say that our method is a form of lazy unfailing completion with no renaming of variables.

However, there are some significant departures from traditional Knuth-Bendix completion procedures, and this is for two reasons. The first reason is that we must ensure termination of the method. The second is that we want to show that the problem is in NP, and this forces us to be much more concerned about efficiency.

Our method can be described in terms of a single transformation on triples of the form $\langle \mathcal{S}, \mathcal{E}, \mathcal{O} \rangle$, where $\mathcal{S}$ is a unifiable set of pairs, $\mathcal{E}$ is a set of equations, and $\mathcal{O}$ is something that will be needed for technical reasons and can be ignored for the present. Starting with an initial triple $\langle \mathcal{S}_0, \mathcal{E}_0, \mathcal{O}_0 \rangle$ initialized using E and u, v (except for $\mathcal{O}$ that must be guessed), if the number of variables in E is m, one considers sequences of transformations

$$\langle \mathcal{S}_0, \mathcal{E}_0, \mathcal{O}_0 \rangle \Rightarrow^+ \langle \mathcal{S}_k, \mathcal{E}_k, \mathcal{O}_k \rangle$$

consisting of at most $k \leq m$ steps. It will be shown that u and v have some rigid E-unifier iff there is some sequence of steps as above such that the special equation $F \doteq T$ is in $\mathcal{E}_k$ and $\mathcal{S}_k$ is unifiable. Then, the most general unifier of $\mathcal{S}_k$ is a rigid E-unifier of u and v.

Roughly speaking, $\mathcal{E}_{k+1}$ is obtained by overlapping equations in $\mathcal{E}_k$ (forming critical pairs), as in unfailing Knuth-Bendix completion procedures, except that no renaming of variables takes place. In order to show that the number of steps can be bounded by m, it is necessary to show that some measure decreases every time an overlap occurs, and there are two difficulties. First, the overlap of two equations may involve the identity substitution when some equation simplifies another one. In this case, the number of variables does not decrease, and no other obvious measure decreases. Second, it is more difficult to handle overlap at variable occurrences than it is in the traditional case, because we are not allowed to form new instances of equations.

The first difficulty can be handled by using a special procedure for reducing a set of (ground) equations. Such a procedure is presented in Gallier et al. [24] and runs in polynomial time (see also [67]). Actually, one also needs a total simplification ordering $\prec$ on ground terms, and a way of orienting equations containing variables, which is the purpose of the mysterious component $\mathcal{O}$. The second difficulty is overcome by noticing that one only needs to consider ground substitutions, that the ordering $\prec$ (on ground terms) can be extended to ground substitutions, and that given any rigid E-unifier θ of u and v, there is always a least rigid E-unifier σ (w.r.t $\prec$) that is equivalent to θ (in a sense to be made precise).

Other complications arise in proving that the method is in NP, in particular, we found it necessary to represent most general unifiers (*mgu*'s) by their triangular form as in Martelli and Montanari [49]. This concept has already been defined in definition 6.2.

The triangular form $T = \{\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\}$ of a substitution σ also defines a substitution, namely $\sigma_T = [t_1/x_1, \ldots, t_k/x_k]$. This substitution is usually different from σ and not idempotent as can be seen from example 6.3. However, this substitution plays a crucial role in our decision procedure because of the following property.

Lemma 8.8 *Given a triangular form $T = \{\langle x_1, t_1\rangle, \ldots, \langle x_k, t_k\rangle\}$ for a substitution σ and the associated substitution $\sigma_T = [t_1/x_1, \ldots, t_k/x_k]$, for every unifier θ of T, $\theta = \sigma_T\,; \theta$.*

An other important observation about σ_T is that even though it is usually not idempotent, at least one variable in $\{x_1, \ldots, x_k\}$ does not belong to $I(\sigma_T)$ (otherwise, condition (1) of the triangular form fails). We will assume that a procedure TU is available, which, given any unifiable term system S, returns a triangular form for an idempotent *mgu* of S, denoted by $TU(S)$. When S consists of a single pair $\langle u, v\rangle$, $TU(S)$ is also denoted by $TU(u, v)$.

One of the major components of the decision procedure for rigid E-unification is a procedure for creating a reduced set of rewrite rules equivalent to a given (finite) set of ground equations. Given a set R of rewrite rules, we say that R *is rigid reduced* iff

(1) No lefthand side of any rewrite rule $l \to r \in R$ is reducible by any rewrite rule in $R - \{l \to r\}$ treated as a ground rule;

(2) No righthand side of any rewrite rule $l \to r \in R$ is reducible by any rewrite rule in R treated as a ground rule.

A procedure for creating a rigid reduced set of rewrite rules equivalent to a given (finite) set of rewrite rules was first presented in Gallier et al. [24] and runs in polynomial time. However, due to the possibility that variables may occur in the equations, we have to make some changes to this procedure. Roughly speaking, given a "guess" $\mathcal{O}$ (a preorder which we call an *order assignment*) of the ordering among all subterms of the terms in a set of equations E, we can run the reduction procedure R on E and $\mathcal{O}$ to produce a reduced rewrite system $R(E, \mathcal{O})$ equivalent to E, and whose orientation is dictated by the preorder $\mathcal{O}$. The precise definition of an order assignment $\mathcal{O}$ is too involved to be reproduced here, but this is not essential anyway. All we need to know is that we have an algorithm R such that, given a set E of equations and an order-assignment $\mathcal{O}$, a rigid-reduced set of rewrite rules $R(E,\mathcal{O})$ is returned, the rules in $R(E,\mathcal{O})$ being oriented by $\mathcal{O}$. We are now ready to define a procedure for finding rigid E-unifiers.

This method uses the reduction procedure just discussed, and a single transformation on certain systems defined next. First, the following definition is needed.

Definition 8.9 Given a set E of equations and some equation $l \doteq r$, the set of equations obtained from E by deleting $l \doteq r$ and $r \doteq l$ from E is denoted by $(E - \{l \doteq r\})^{\dagger}$. Formally, we let $(E - \{l \doteq r\})^{\dagger} = \{u \doteq v \mid u \doteq v \in E,\ u \doteq v \neq l \doteq r,\ \text{and } u \doteq v \neq r \doteq l\}$.

Definition 8.10 Let $\prec$ be a total simplification ordering on ground terms. We shall be considering finite sets of equations of the form $\mathcal{E} = \mathcal{E}_{\Sigma} \cup \{eq(u,v) \doteq F,\ eq(z,z) \doteq T\}$,[10] where $\mathcal{E}_{\Sigma}$ is a set of equations over $T_{\Sigma}(\mathcal{X})$, and $u, v \in T_{\Sigma}(\mathcal{X})$. We define a transformation on systems of the form $\langle \mathcal{S}, \mathcal{E}, \mathcal{O}\rangle$, where $\mathcal{S}$ is a term system, $\mathcal{E}$ a set of equations as above, and $\mathcal{O}$ an order assignment:

$$\langle \mathcal{S}_0,\ \mathcal{E}_0,\ \mathcal{O}_0\rangle \Rightarrow \langle \mathcal{S}_1,\ \mathcal{E}_1,\ \mathcal{O}_1\rangle,$$

where $l_1 \doteq r_1,\ l_2 \doteq r_2 \in \mathcal{E}_0 \cup \mathcal{E}_0^{-1}$, either l_1/β is *not* a variable or the equation $l_2 \doteq r_2$ is degenerate,[11] $l_1/\beta \neq l_2$, $TU(l_1/\beta, l_2)$ represents an *mgu* of l_1/β and l_2 in triangular form,[12] $\sigma = [t_1/x_1, \ldots, t_p/x_p]$ where $TU(l_1/\beta, l_2) = \{\langle x_1, t_1\rangle, \ldots, \langle x_p, t_p\rangle\}$,

$$\mathcal{E}_1' = \sigma((\mathcal{E}_0 - \{l_1 \doteq r_1\})^{\dagger} \cup \{l_1[\beta \leftarrow r_2] \doteq r_1\}),$$

$\mathcal{O}_1$ is an order assignment on $\mathcal{E}_1'$ compatible with $\mathcal{O}_0$, $\mathcal{S}_1 = \mathcal{S}_0 \cup TU(l_1/\beta, l_2)$, and $\mathcal{E}_1 = R(\mathcal{E}_1', \mathcal{O}_1)$.

Observe that $\sigma(l_1[\beta \leftarrow r_2] \doteq r_1)$ looks like a critical pair of equations in $\mathcal{E}_0 \cup \mathcal{E}_0^{-1}$, but it is not.This is because a critical pair is formed by applying the *mgu* of l_1/β and l_2 to $l_1[\beta \leftarrow r_2] \doteq r_1$, but $[t_1/x_1, \ldots, t_p/x_p]$ is usually not a *mgu* of l_1/β and l_2. It is the composition $[t_1/x_1]\,; \ldots ;[t_p/x_p]$ that is a *mgu* of l_1/β and l_2. The reason for not applying the *mgu* is that by repeated applications of this step, exponential size terms could be formed, and it would not be clear that the decision procedure is in NP. We have chosen an approach of "lazy" (or delayed) unification. Also note that we use the rigid reduced system $R(\mathcal{E}_1', \mathcal{O}_1)$ rather than $\mathcal{E}_1'$, and so, a transformation step is defined only if R does not fail. The method for finding E-unifiers is then is the following.

[10] eq, T, F are some new symbols not occurring in E, u, v.

[11] An equation $x \doteq v$ is degenerate if x is a variable and $x \notin Var(v)$.

[12] Note that we are requiring that l_1/β and l_2 have a *nontrivial* unifier. The triangular form of *mgus* is important for the NP-completeness of this method.

Definition 8.11 (Method) Let $E_{u,v} = E \cup \{eq(u,v) \doteq F,\ eq(z,z) \doteq T\}$, $\mathcal{O}_0$ an order assignment on $E_{u,v}$, $\mathcal{S}_0 = \emptyset$, $\mathcal{E}_0 = R(E_{u,v}, \mathcal{O}_0)$, m the total number of variables in $\mathcal{E}_0$, and $V = Var(E) \cup Var(u,v)$. For any sequence

$$\langle \mathcal{S}_0, \mathcal{E}_0, \mathcal{O}_0 \rangle \Rightarrow^+ \langle \mathcal{S}_k, \mathcal{E}_k, \mathcal{O}_k \rangle$$

consisting of at most m transformation steps, if $\mathcal{S}_k$ is unifiable and $k \leq m$ is the first integer in the sequence such that $F \doteq T \in \mathcal{E}_k$, return the substitution $\theta_{\mathcal{S}_k}|_V$, where $\theta_{\mathcal{S}_k}$ is the *mgu* of $\mathcal{S}_k$ (over $T_\Sigma(\mathcal{X})$).

Example 8.12 Let E be the set of equations $E = \{fa \doteq a,\ ggx \doteq fa\}$, and $\langle u, v \rangle = \langle gggx, x \rangle$. We have

$$E_{u,v} = \{fa \doteq a,\ ggx \doteq fa,\ eq(gggx, x) \doteq F,\ eq(z,z) \doteq T\}.$$

The congruence closure Π of $E_{u,v}$ has three nontrivial classes $\{a, fa, ggx\}$, $\{eq(gggx,x), F\}$, and $\{eq(z,z), T\}$. Let $\mathcal{O}_0$ be the order assignment on $E_{u,v}$ such that

$$\begin{aligned} &T \prec_{\mathcal{O}_0} eq(gggx, x),\\ &F \prec_{\mathcal{O}_0} eq(z,z),\\ &a \prec_{\mathcal{O}_0} fa \prec_{\mathcal{O}_0} ggx, \end{aligned}$$

the least elements of classes being ordered in the order of listing of the classes. We have $\mathcal{S}_0 = \emptyset$, and the reduced system $\mathcal{E}_0 = R(E_{u,v}, \mathcal{O}_0)$ is

$$\mathcal{E}_0 = \{fa \doteq a,\ ggx \doteq a,\ eq(ga, x) \doteq F,\ eq(z,z) \doteq T\}.$$

Note that there is an overlap between $eq(ga,x) \doteq F$ and $eq(z,z) \doteq T$ at address ϵ in $eq(ga,x)$, and we obtain the triangular system $\{\langle x, ga\rangle, \langle z, ga\rangle\}$ and the new equation $F \doteq T$. Thus, we have

$$\langle \mathcal{S}_0, \mathcal{E}_0, \mathcal{O}_0 \rangle \Rightarrow \langle \mathcal{S}_1, \mathcal{E}_1, \mathcal{O}_1 \rangle,$$

where $\mathcal{S}_1 = \{\langle x, ga\rangle, \langle z, ga\rangle\}$

$$\mathcal{E}_1' = \{fa \doteq a,\ ggga \doteq a,\ eq(ga, ga) \doteq F,\ F \doteq T\},$$

and $\mathcal{O}_1$ is the restriction of $\mathcal{O}_0$ to the subterms in $\mathcal{E}_1'$. After reducing $\mathcal{E}_1'$, we have

$$\mathcal{E}_1 = \{fa \doteq a,\ ggga \doteq a,\ eq(ga, ga) \doteq T,\ F \doteq T\}.$$

Since $F \doteq T \in \mathcal{E}_1$ and $\mathcal{S}_1$ is unifiable, the restriction $[ga/x]$ of the *mgu* $[ga/x, ga/z]$ of $\mathcal{S}_1$ to $Var(E) \cup Var(u,v) = \{x\}$ is a rigid E-unifier of $gggx$ and x.

The following major results are proved in Gallier, Narendran, Plaisted, and Snyder [27].

Theorem 8.13 *The procedure given by definition 8.11 is a decision procedure for rigid E-unification. Furthermore, it belongs to NP.*

The soundness and completeness of the method are subsumed by the following result.

Theorem 8.14 *Let E be a set of equations over $T_\Sigma(\mathcal{X})$, u, v two terms in $T_\Sigma(\mathcal{X})$, m the number of variables in $E \cup \{u, v\}$, and $V = Var(E) \cup Var(u, v)$. There is a finite complete set of rigid E-unifiers for u and v given by the set*

$$\{\theta_{\mathcal{S}_k}|_V \;\mid \langle \mathcal{S}_0, \mathcal{E}_0, \mathcal{O}_0 \rangle \Rightarrow^+ \langle \mathcal{S}_k, \mathcal{E}_k, \mathcal{O}_k \rangle,\; k \le m\},$$

for any order assignment $\mathcal{O}_0$ on $E_{u,v}$, with $\mathcal{S}_0 = \emptyset$, $\mathcal{E}_0 = R(E_{u,v}, \mathcal{O}_0)$, and where $\mathcal{S}_k$ is unifiable, $F \doteq T \in \mathcal{E}_k$, $F \doteq T \notin \mathcal{E}_i$ for all i, $0 \le i < k$, and $\theta_{\mathcal{S}_k}$ is the mgu of $\mathcal{S}_k$ over $T_\Sigma(\mathcal{X})$.

Thus, we note another major difference between general E-unification and rigid E-unification. In rigid E-unification, there is always a finite complete set of (rigid) E-unifiers.

The above results have been improved by Isakowitz [37] and by Choi and Gallier [17]. Isakowitz has shown that order assignments can be dispensed with if a different reduction procedure is used. Isakowitz also studied the extension of rigid E-unification to order-sorted logic, and proved results analogous to those presented here for some subclasses of equations. Choi and Gallier have obtained a more direct proof of the NP-completeness of rigid E-unification that also avoids order assignments. This new proof is more algebraic and uses some key ideas from Kozen [45].

9 Conclusion and Directions For Further Research

We surveyed two methods for automated theorem proving, the method of matings for languages without equality, and the method of equational matings, for languages with equality. We also surveyed various unification procedures associated with theorem proving methods based on matings. These include standard unification and rigid E-unification. The crucial property of these unification methods is that they are decidable. However, their complexity is very different: standard unificatin can be performed in linear-time, but rigid E-unification is NP-complete.

An area of research that remains wide open is the study of efficient implementations of equational matings and rigid E-unification. This is a difficult problem, since rigid E-unification is NP-complete, and one needs to isolate interesting special classes of formulae for which tractable algorithms can be found. On a more theoretical level, it would be interesting to study the generalization of linear logic including equations. We conjecture that this will lead naturally to rigid E-unification. Finally, investigating whether the method of matings can be generalized either to order-sorted logic or to higher-order logic, remains to be done.

10 References

[1] Aho, A., Sethi, R., and Ullman, J. *Compilers, Principles, Techniques, and Tools*, Addison Wesley (1986).

[2] Aït-Kaci, H. A lattice theoretic approach to computation based on a calculus of partially ordered type structures, Ph.D. thesis. Department of Computer and Information Science, University of Pensylvania, PA (1984).

[3] Aït-Kaci, H. An algebraic semantics approach to the effective resolution of type equations. *Theoretical Computer Science* 45, pp. 293-351 (1986).

[4] Andrews, P. Theorem Proving via General Matings. *J.ACM* 28(2), 193-214, 1981.

[5] Andrews, P. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof.* Academic Press, New York, 1986.

[6] Andrews, P.B., D. Miller, E. Cohen, F. Pfenning, "Automating Higher-Order Logic," *Contemporary Mathematics* 29, 169-192, 1984.

[7] Bachmair, L., *Proof Methods for Equational Theories*, Ph.D thesis, University of Illinois, Urbana Champaign, Illinois (1987).

[8] Bachmair, L., *Canonical Equational Proofs*, Research Notes in Theoretical Computer Science, Wiley and Sons, 1989.

[9] Bachmair, L., Dershowitz, N., and Plaisted, D., "Completion without Failure," *Resolution of Equations in Algebraic Structures*, Vol. 2, Aït-Kaci and Nivat, editors, Academic Press, 1-30 (1989).

[10] Bachmair, L., Dershowitz, N., and Hsiang, J., "Orderings for Equational Proofs," In *Proc. Symp. Logic in Computer Science*, Boston, Mass. (1986) 346-357.

[11] Bibel, W. Tautology Testing With a Generalized Matrix Reduction Method, *TCS* 8, pp. 31-44, 1979.

[12] Bibel, W. On Matrices With Connections, *J.ACM* 28, pp. 633-645, 1981.

[13] Bibel, W. *Automated Theorem Proving.* Friedr. Vieweg & Sohn, Braunschweig, 1982.

[14] Boudet, A., Jouannaud, J.-P., and Schmidt-Schauss, M. Unification in Boolean Rings and Abelian Groups. *Journal of Symbolic Computation* 8(5), pp. 449-478 (1989).

[15] Bürckert, H., Herold, A., and Schmidt-Schauss, M. On equational theories, unification, and (un)decidability. Special issue on Unification, Part II, *Journal of Symbolic Computation* 8(1 & 2), 3-50 (1989).

[16] Bürckert, H., Matching–A Special Case of Unification? *Journal of Symbolic Computation* 8(5), pp. 523-536 (1989).

[17] Choi, J., and Gallier, J.H. A simple algebraic proof of the NP-completeness of rigid E-unification, in preparation (1990).

[18] Church, A., "A note on the Entscheidungsproblem", *JSL* 1 (1936) 40-41, corrections, 101-102.

[19] De Champeaux, D., "About the Paterson-Wegman linear unification," *Journal of Computer and System Sciences*, 32(1) (1986) 79-90.

[20] Dershowitz, N,. "Termination of Rewriting," Journal of Symbolic Computation 3 (1987) 69-116.

[21] Downey, Peter J., Sethi, Ravi, and Tarjan, Endre R. "Variations on the Common Subexpressions Problem." *J.ACM* 27(4), 758-771, 1980.

[22] Gallier, J.H. *Logic for Computer Science: Foundations of Automatic Theorem Proving*, Harper and Row, New York (1986).

[23] Gallier, J.H., Raatz, S., and Snyder, W., "Theorem Proving using Rigid E-Unification: Equational Matings," *LICS'87*, Ithaca, New York (1987) 338-346.

[24] Gallier, J.H., Narendran, P., Plaisted, D., Raatz, S., and Snyder, W., "Finding canonical rewriting systems equivalent to a finite set of ground equations in polynomial time," submitted to *J.ACM* (1987).

[25] Gallier, J.H., Narendran, P., Plaisted, D., and Snyder, W., "Rigid E-unification is NP-complete," *LICS'88*, Edinburgh, Scotland, July 5-8, 1988, 218-227.

[26] Gallier, J.H., Raatz, S, and Snyder, W. Rigid E-Unification and its Applications to Equational Matings. *Resolution of Equations in Algebraic Structures*, Vol. 1, Aït-Kaci and Nivat, editors, Academic Press, 151-216 (1989).

[27] Gallier, J.H., Narendran, P., Plaisted, D., and Snyder, W. Rigid E-Unification: NP-completeness and Applications to Theorem Proving. Special issue of *Information and Computation* 87(1/2), 129-195 (1990).

[28] Gallier, J.H., Narendran, P., Raatz, S., and Snyder, W. Theorem Proving Using Equational Matings and Rigid E-Unification. To appear in *J.ACM*, pp. 62 (1990).

[29] Girard, J.Y., "Linear Logic," *Theoretical Computer Science* 50:1 (1987) 1-102.

[30] Gödel, Kurt. Die Vollstandigkeit der Axiome des Logischen Funktionenkalküls, *Monatsh. Math. Phys.* 37, 349-360 (1930), translated in Gödel [31], 44-123 (1986).

[31] Gödel, Kurt. *Collected Works, Vol. I, Publications 1929-1936*, Edited by S. Feferman, J. Dawson, S. Kleene, G. Moore, R. Solovay, and J. van Heijenoort, Oxford University Press (1986).

[32] Herbrand, J., "Sur la Théorie de la Démonstration," in *Logical Writings*, W.D. Goldfarb, ed., Harvard University Press (1971).

[33] Hindley, J., and Seldin, J., *Introduction to Combinators and Lambda Calculus*, Cambridge University Press (1986).

[34] Huet, G. *Constrained Resolution: A Complete Method for Higher-Order Logic*, Ph.D. thesis, Case Western Reserve University (1972).

[35] Huet, G., *Résolution d'Equations dans les Langages d'Ordre* $1, 2, \ldots, \omega$, Thèse d'Etat, Université de Paris VII (1976).

[36] Huet, G., "Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems," JACM 27:4 (1980) 797-821.

[37] Isakowitz, T. Theorem Proving Methods For Order-Sorted Logic, Ph.D. thesis. Department of Computer and Information Science, University of Pensylvania (1989).

[38] Jouannaud, J.-P., and Kirchner, C. Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification. Technical Report, University of Paris Sud (1989).

[39] Kirchner, C., *Méthodes et Outils de Conception Systematique d'Algorithmes d'Unification dans les Theories Equationnelles*, Thèse d'Etat, Université de Nancy I (1985).

[40] Kfoury, A.J., J. Tiuryn, and P. Urzyczyn, "An analysis of ML typability", submitted (a section of this paper will appear in the proceedings of CAAP 1990 under the title "ML typability is DEXPTIME-complete").

[41] Kfoury, A.J., J. Tiuryn, and P. Urzyczyn, "The undecidability of the semi-unification problem", Proceedings of STOC (1990).

[42] Kfoury, A.J., and J. Tiuryn, "Type Reconstruction in Finite-Rank Fragments of the Polymorphic Lambda Calculus," LICS'90, Philadelpha, PA.

[43] Knight. K. "A Multidisciplinary Survey," *ACM Computing Surveys*, Vol. 21, No. 1, pp. 93-124 (1989).

[44] Knuth, D.E. and Bendix, P.B., "Simple Word Problems in Univeral Algebras," in *Computational Problems in Abstract Algebra*, Leech, J., ed., Pergamon Press (1970).

[45] Kozen, D., "Complexity of Finitely Presented Algebras," Technical Report TR 76-294, Department of Computer Science, Cornell University, Ithaca, New York (1976).

[46] Kozen, D., "Positive First-Order Logic is NP-Complete," IBM Journal of Research and Development, 25:4 (1981) 327-332.

[47] Lassez, J.-L., Maher, M., and Marriot, K. Unification Revisited. *Foundations of Deductive Databases and Logic Programming*, J. Minker, editor, Morgan-Kaufman, pp. 587-625 (1988).

[48] Manna, Zohar. *Mathematical Theory of Computation.* McGraw-Hill (1974).

[49] Martelli, A., Montanari, U., "An Efficient Unification Algorithm," ACM Transactions on Programming Languages and Systems, 4:2 (1982) 258-282.

[50] Meseguer, J., Goguen, J. A., and Smolka, G. Order-Sorted Unification. *Journal of Symbolic Computation* 8(4), pp. 383-413 (1989).

[51] Miller, D., *Proofs in Higher-Order Logic*, Ph.D. thesis, Carnegie-Mellon University, 1983.

[52] Miller, D. A. Expansion Trees and Their Conversion to Natural Deduction Proofs. In *7th International Conference on Automated Deduction, Napa, CA*, edited by R.E. Shostak, L.N.C.S, No. 170, New York: Springer Verlag, 1984.

[53] Miller, D. A compact Representation of Proofs. *Studia Logica* 4/87, pp. 347-370 (1987).

[54] Milner, R. A theory of type polymorphism in programming. *J. Comput. Sys. Sci.* 17, pp. 348-375 (1978).

[55] Nelson G. and Oppen, D. C. Fast Decision Procedures Based on Congruence Closure. *J. ACM* 27(2), 356-364, 1980.

[56] Paterson, M.S., Wegman, M.N., "Linear Unification," Journal of Computer and System Sciences, 16 (1978) 158-167.

[57] Pfenning, F., *Proof Transformations in Higher-Order Logic*, Ph.D. thesis, Department of Mathematics, Carnegie Mellon University, Pittsburgh, Pa. (1987).

[58] Plotkin, G., "Building in Equational Theories," Machine Intelligence 7 (1972) 73-90.

[59] Rémy, Didier, "Algèbres Touffues. Application au typage polymorphique des objects enregistrements dans les languages fonctionnels." Thèse, Université Paris VII, 1990.

[60] Rémy, Didier, "Records and variants as a natural extension in ML," *Sixteenth ACM Annual Symposium on Principles of Programming Languages*, Austin Texas, 1989.

[61] Robinson, J.A., "A Machine Oriented Logic Based on the Resolution Principle," JACM 12 (1965) 23-41.

[62] Siekmann, J. H. Unification Theory, Special Issue on Unification, Part I, *Journal of Symbolic Computation* 7(3 & 4), pp. 207-274 (1989).

[63] Shieber, S. *An Introduction to Unification-Based Approaches to Grammar.* CSLI Lecture Notes Series, Center for the study of Language and Information, Stanford, CA (1986).

[64] Skolem, Thoralf. Über die Mathematische Logik, *Norsk matematisk tidsskrift* 10, 125-142 (1928). translated in van Heijenoort [70], 508-524 (1967).

[65] Schmidt-Schauss, M. Unification in a Combination of Arbitrary Disjoint Equational Theories. Special issue on Unification, Part II, *Journal of Symbolic Computation* 8(1 & 2), 51-99 (1989).

[66] Snyder, W. Complete Sets of Transformations for General Unification, Ph.D. thesis. Department of Computer and Information Science, University of Pensylvania, PA (1988).

[67] Snyder, W., "Efficient Ground Completion: A Fast Algorithm for Generating Reduced Ground Rewriting Systems from a Set of Ground Equations," RTA'89, Chapel Hill, NC (journal version submitted for publication).

[68] Snyder, W. *The Theory of General Unification.* Birkhauser Boston, Inc. (in preparation).

[69] Szabo, P. *Unifikationstheorie erster Ordnung*, Ph.D. thesis, Universität Karlsruhe (1982).

[70] van Heijenoort, Jean (editor). *From Frege to Gödel. A Source Book in Mathematical Logic, 1879-1931*, Harvard University Press (1967).

[71] Yelick, K. Unification in combinations of collapse-free regular theories. *Journal of Symbolic Computation* 3(1 & 2), pp. 153-182 (1987).