General Algorithms for Permutations in Equational Inference [*]

Jürgen Avenhaus
Universität Kaiserslautern
Fachbereich Informatik
D-67663 Kaiserslautern
e-mail: avenhaus@informatik.uni-kl.de

David A. Plaisted
Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175
e-mail: plaisted@cs.unc.edu

Abstract

We study some general algorithms for processing permutations and permutation groups and consider their application to equational reasoning and term-rewriting systems. We also present some complexity results for particular equational consequence problems related to permutations.

1 Introduction

A common problem in applications of mechanized reasoning is to determine if an equation $s = t$ is a logical consequence of a set E of equations, that is, to determine whether $E \models s = t$. Even in the more general context of first-order theorem proving, deduction using equations plays an important role. A problem that frequently arises in such theorem-proving applications of term-rewriting systems and equational reasoning is that many permutations of a given term or equation are often produced. For example, we may obtain many equations like $f(x, f(y, f(z, w))) = f(w, f(z, f(y, x)))$, where the left and right hand sides are the same except for a renaming of the variables. We can view such equations as permutations of the variables x, y, z, w. The number of such

[*]This research was partially supported by the National Science Foundation under grant CCR-9108904

equations can be very large, and they can substantially slow down the search process, since many inferences must be performed over and over again that differ only in a permutation of variables. Using such permutations, we can also derive many equations of the form $s = t_i$ where the t_i are identical except for a renaming of the variables, and s is an arbitrary term. For example, we may derive equations like $g(x,y,z,w) = f(x, f(y, f(z,w)))$, $g(x,y,z,w) = f(w, f(z, f(y,x)))$, and so on. This is therefore another kind of set of equations that can arise and cause problems for equational inference. The equations $s = t_i$ themselves do not involve permutations of the variables, but the equations are in some sense permutations of each other. We would like to develop methods to handle them efficiently, as well. In addition, we can obtain terms in which subterms are permuted, rather than variables, as above. For example, we might obtain many equations of the form $f(u_1, u_2, u_3) = f(u_2, u_3, u_1)$, and similar equations where the non-variable subterms u_1, u_2, and u_3 are permuted. Even one fairly large equation can produce thousands of other equations through these various kinds of permutations of subterms. For example, consider an equation

$$x_1 * x_2 * x_3 + x_2 * x_3 * x_4 + x_3 * x_4 * x_5 = 1$$

where the infix operators $*$ and $+$ are associative and commutative. There are three products of three terms; each product can independently be permuted in 6 ways, leading to 216 combinations. One example would be

$$x_3 * x_2 * x_1 + x_2 * x_3 * x_4 + x_4 * x_3 * x_5 = 1$$

In addition, the three products can be permuted among themselves in 6 ways, for example as follows:

$$x_2 * x_3 * x_4 + x_3 * x_4 * x_5 + x_1 * x_2 * x_3 = 1$$

leading to a total of 1296 ways of writing essentially the same polynomial equation. Of course, for polynomials having many more terms and products, this number can easily become astronomical. This is unsatisfactory, because there are simple ways to deal with large polynomials that are efficient in practice. This problem can be dealt with using constraints or AC-unification, but we have found these to be inefficient in practice in some cases, and it would be desirable to develop additional approaches.

Such permutations arise in the presence of associative and commutative operators. They may also arise in other contexts. We may obtain rotations from an equation of the form $f(x_1, \ldots, x_n) = f(x_2, x_3, \ldots, x_n, x_1)$. Or, a function symbol may have two disjoint sets of arguments, each of which can be permuted independently in an arbitrary manner.

One of the most common methods to deal with this problem is to use specialized unification algorithms for various equational theories, such as AC-theories. This has led to a large number of specialized unification algorithms for various theories, and a large number of results concerning how these unification algorithms can be combined. However, we would like to develop general methods for dealing directly with subterm permutations in term-rewriting theorem proving, regardless of which set of equations produce them, instead of separately treating various equational theories that tend to produce subterm permutations. We would like to find efficient methods for representing and processing terms that are permutations of one another, to avoid storing and processing them separately. Also, we hope in this way to avoid the large (double exponential) number of unifiers that can be produced by associative-commutative unification [Dom92]. Another approach to AC theories and other equational theories is to use constraints, but we have found this to be inefficient in practice. Yet another approach that we have not tried would be to develop a general E-unification theory for equations that produce subterm permutations. For discussions of the general area of equational reasoning, see [DJ90, Pla93, Klo92].

Our goal, then, is to handle E-equivalent terms efficiently, where E is a set of equations. The idea of our approach is not to deal with entire E-equivalence classes of terms, as is done in classical E-unification, but rather to deal with a finer partition (i.e., smaller classes) that have convenient properties permitting efficient algorithms to be applied. This may mean that more equations/clauses need to be retained, but their processing may be faster. Also, our method is general, and so may be able to exploit algorithmic efficiencies not covered in known equational unification algorithms.

We now give a general outline of our approach. Suppose we are given a set E of equations and a set A of first-order formulas (typically clauses), and wish to determine if $A \cup E$ is satisfiable relative to equality. Our approach is to construct pairs (t, G), where t is a term and G is a group of mappings from terms to terms. We denote by $S(t, G)$ the set of terms that can be obtained by the action of G on the term t. The group G satisfies the property that if $u \in S(t, G)$, then $E \models t = u$, that is, $S(t, G)$ is a subset of the E-equivalence class of t. Our goal is to use $S(t, G)$ to approximate the E-equivalence class of t, and deal with $S(t, G)$ instead of with the E-equivalence class. The advantage of this is that we can choose G so that processing the class $S(t, G)$ can take advantage of efficient algorithms that have been developed for processing permutations. We try to find groups G that will approximate the E-equivalence classes as closely as possible, while still permitting such efficient algorithms to be applied. We also try to maintain disjointness, which means that if we have pairs (t_1, G_1) and (t_2, G_2), then $S(t_1, G_1) \cap S(t_2, G_2) = \phi$. This helps to reduce the number of pairs needed. One possible application of this idea is to simulate unfailing completion [BDP89]. Instead of dealing with single equations and formulas, we propose to deal with the sets $S(t, G)$ directly, thereby avoiding the necessity of handling all of the elements of $S(t, G)$ one by one. For this purpose, we need a way to extend the $S(t, G)$ notation to describe sets of equations or clauses. To this end, we propose to represent the set $\{v_1 = v_2 : v_1 \in S(t_1, G_1), v_2 \in S(t_2, G_2)\}$ of equations by the pair $(t_1 = t_2, \{(t_1, G_1), (t_2, G_2)\})$. We note that all such equations $v_1 = v_2$ are logical consequences of E and the equation $t_1 = t_2$. Similary, if C is a clause and $t_1, \cdots, t_n$ are its maximal subterms, then we can represent by $(C[t_1, \cdots, t_n], \{(t_1, G_1), \cdots, (t_n, G_n)\})$ the set of clauses $\{C[v_1, \cdots, v_n] : v_1 \in S(t_1, G_1), \cdots, v_n \in S(t_n, G_n)\}$. We propose to perform operations on these sets of equations or clauses corresponding to the operations that would have been done by unfailing completion on $A \cup E$, but gaining efficiency by taking advantages of the regularity in the structure of the sets $S(t, G)$. We develop this general framework, and also look for special cases where even more efficient algorithms can be applied.

We now suggest the kinds of efficiencies that may be gained by known algorithms for permutations. We know by results of [Fro89] that certain equivalence problems relative to a set of permutations, can be solved in polynomial time. For example, we say that an equation of the form $f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ where the variables x_i are all distinct and the y_i are a permutation of the x_i, is a *flat permutative* equation or a *flat permutation.* If E is a set of flat permutative equations then we call E a *flat permutative theory.* Consider the problem of deciding whether $E \models s = t$ where E is a flat permutative theory and $s = t$ is a flat permutation. We call this the *flat permutative consequence problem.*

Theorem 1.1 *The flat permutative consequence problem has a polynomial time solution.*

Proof. See [Fro89]. □

Corollary 1.2 *If E is a flat permutative theory and s is a ground term of the form $f(a_1, a_2, \ldots, a_n)$ and t is of the form $f(b_1, b_2, \ldots, b_n)$ where the a_i are distinct constant symbols and the b_i are a permutation of the a_i, then the problem of deciding whether $E \models s = t$ has a polynomial time solution.*

We note that if one uses traditional equational reasoning methods on the flat permutative consequence problem, the solution may take exponential time, since a permutation group over n elements can have a number of elements exponential in n. This motivates the search for other efficient algorithms relating to subterm permutations and term rewriting. We will show later that one advantage of the present approach is that it can sometimes detect the existence of long rewriting proofs without actually constructing them, thereby possibly gaining efficiency. The algorithm for theorem 1.1 is based on the following result, from [FHL80]:

Theorem 1.3 *Suppose $g_1, g_2, \ldots, g_n$ are permutations and g is a permutation. Then the problem of deciding whether g is in the group generated by $g_1, g_2, \ldots, g_n$ is solvable in polynomial time.*

We call this problem, the *permutation group membership problem.*

1.1 Terminology

We first develop some terminology. A *term* is a well-formed expression composed of function symbols, constant symbols, and variables, for example $f(a, g(x, y))$. We specify positions of terms as usual as sequences of integers. If α and β are positions, then we say $\alpha \leq \beta$ if α is a prefix of β, and $\alpha < \beta$ if α is a proper prefix of β. If neither of α or β is a prefix of the other, then we say that α and β are *independent* or *disjoint*. We write $r \equiv s$ to indicate that the terms r and s are identical. If α is a position and t is a term, we write $t|_\alpha$ or $\alpha(t)$ for the subterm of t at position α, defined by $\alpha(t) \equiv t$ if α is the empty sequence, otherwise $i\alpha(f(t_1, \ldots, t_m)) \equiv \alpha(t_i)$. Often it is easier to speak in terms of *contexts* instead of positions. A *context* is a term with occurrences of $\square$ in it. For example, $f(\square, g(a, \square))$ is a context. A $\square$ by itself is also a context. If n is an integer then an *n-context* is a term with n occurrences of $\square$. If t is an n-context and $m \leq n$ then $t[t_1, \ldots, t_m]$ represents t with the leftmost m occurrences of $\square$ replaced by the terms $t_1, \ldots, t_m$, respectively. We call such a term $t[t_1, \ldots, t_m]$ an *instance* or *context instance* of the context t. Note that the t_i may also be contexts. We write $u \geq t$ if u is a context instance of context t. We write $u > t$ if $u \geq t$ and $u \not\equiv t$. If r is a 1-context and s is an n-context for $n \geq 1$, then $r[s][t] \equiv r[s[t]]$, by a simple argument (both replace the leftmost box in $r[s]$ by t). Also, if r is an n-context and $m \leq n$ and the terms s_i are 0-contexts, then $r[s_1, \ldots, s_n] \equiv r[s_1][s_2] \ldots [s_n]$. If t is a 0-context, then we define t^α for a position α to be t with the term at the α position replaced by $\square$. Thus t^i has $\square$ for the i^{th} subterm and $t^i[u]$ is t with the i^{th} subterm replaced by u. If $\mathcal{P}$ is a set of n independent positions and t is an 0-context, then $t^{\mathcal{P}}$ is the n-context in which $t^{\mathcal{P}}|_\alpha \equiv \square$ for all α in $\mathcal{P}$, and $t^{\mathcal{P}}$ is identical to t otherwise.

Two terms r and s are *variants* if they are instances of each other; that is, r may be expressed as $r[x_1, \ldots, x_n]$ and s may be expressed as $r[y_1, \ldots, y_n]$ where the x_i and y_i are variables, $x_i \equiv x_j$ iff $y_i \equiv y_j$, and all occurrences of the variables x_i and y_i are explicitly listed. Thus the terms $f(x, f(y, z))$ and $f(u, f(x, w))$ are variants. If E is a set of equations, Θ is a substitution, and u is a 1-context, then we say that $u[r\Theta] \rightarrow_E u[s\Theta]$ if $r = s$ is a member of E such that all variables in s appear also in r. We say $r \leftrightarrow_E s$ if $s \rightarrow_E r$ or $r \rightarrow_E s$. As usual, a superscript of $*$ indicates reflexive and transitive closure. An equation $r = s$ is a *leaf permutation* or *leaf permutative* if r and s are linear terms that have the same set of variables and are variants of each other. Thus $f(x, f(y, z)) = f(y, f(x, z))$ is leaf permutative, but $f(x, f(y, z)) = f(w, f(x, y))$ is not. A set $\{r_1 = s_1, \ldots, r_n = s_n\}$ of leaf permutative equations is *uniform* if for all i and j, r_i and r_j are variants. Thus the set of three equations

$$f(x, f(y, f(z, w))) = f(y, f(x, f(z, w)))$$

$$f(x, f(y, f(z, w))) = f(x, f(z, f(y, w)))$$

$$f(x, f(y, f(z, w))) = f(x, f(y, f(w, z)))$$

is uniform, and all elements of this set are logical consequences of the associative and commutative equations for f. However, the following set of two equations is not uniform:

$$f(x, f(y, f(z, w))) = f(y, f(z, f(w, x)))$$

$$f(u,v) = f(v,u)$$

This set is not uniform because the terms $f(x, f(y, f(z,w)))$ and $f(u,v)$ are not variants; $f(x, f(y, f(z,w)))$ is an instance of $f(u,v)$, but $f(u,v)$ is not an instance of $f(x, f(y, f(z,w)))$. If a set $\{r_1 = s_1, \ldots, r_n = s_n\}$ is uniform, then we call a term s that is a variant of r_i (or s_i) a *redex for* $\{r_1 = s_1, \ldots, r_n = s_n\}$. The term $f(z, f(w, f(y,x)))$ is a redex for the uniform set given above.

2 Subterm Permutation Groups

We first discuss subterm permutation groups, and then show how more than one subterm permutation group can be combined into a stratified subterm permutation group. Subterm permutation group are a particular class of groups of mappings from terms to terms which generate subsets of the E-equivalence class of a term. Formally, we define a *subterm permutation group* G as a 1-context $u[\ ]$ together with a uniform set E_G of leaf permutative equations. We denote G by the ordered pair $(u[\ ], E_G)$. The equations $r = s$ in E_G are converted into term mappings according to the following definition, which will often be referred to in the sequel:

Definition 2.1 *Suppose that r and s are terms and u is a 1-context. Suppose all variables in s appear also in r. Then the mapping $\phi_{r,s,u}$ from terms to terms is defined by*

$$(\forall\Theta)\{(t \equiv u[r\Theta]) \supset (\phi_{r,s,u}(t) \equiv u[s\Theta])\}$$
$$\{(\neg\exists\Theta)(t \equiv u[r\Theta])\} \supset (\phi_{r,s,u}(t) \equiv t)$$

Often we write $\phi_{r,s,u}$ as $\phi_{r,s}$ when u is apparent from context, or even as ϕ.

The *domain* of the subterm permutation group $(u[\ ], E_G)$ consists of terms of the form $u[r\Theta]$ for various substutions Θ, where r is an E_G-redex. Each mapping $\phi_{r,s}$ for $r = s$ in E_G maps terms $u[r\Theta]$ in the domain of G to terms $u[s\Theta]$, for all substitutions Θ. Terms not of the form $u[r\Theta]$ map to themselves; this is not relevant here, but will be convenient later. We let $\Phi(G)$ be the set of such $\phi_{r,s}$, for equations $r = s$ in E_G, and refer to $\Phi(G)$ as the *generators* of G. We also write $\Phi(G)$ as $\Phi(u[\ ], E_G)$. Note that $u[s\Theta]$ is also in the domain of G, because the terms r and s are variants. Thus the mappings ϕ can be composed. These mappings $\phi_{r,s}$ then generate a group under the operation of function composition. Thus the term "subterm permutation group." We define the composition operator $\cdot$ on functions so that $(\phi \cdot \phi')(t) \equiv \phi(\phi'(t))$. We let $\Phi^*(G)$ denote the group generated by $\Phi(G)$ under the operation of function composition. For computational purposes, it is easier to store just the generators $\Phi(G)$ rather than the whole group $\Phi^*(G)$. We define $S(t, G)$ to be the set of terms that may be obtained from t by a composition of such mappings ϕ in $\Phi(G)$. Thus $S(t, G) = \{\phi(t) : \phi \in \Phi^*(G)\}$. Note that if $t' \in S(t, G)$, then $S(t', G) = S(t, G)$. Also, $S(t, G)$ is finite since its cardinality is bounded by that of a finite permutation group.

We also require that each equation $r = s$ be a logical consequence of E, that is, $E \models r = s$. An example was given earlier of a set of three uniform equations, all consequences of an associative and commutative axiom for f. This implies that if $t' \in S(t,G)$, then $E \models t = t'$, by a simple proof. However, it is possible that there exists a term u such that $E \models t = u$ but $u \notin S(t,G)$.

Example 2.2 As an example of a subterm permutation group G, let E_G be the set of three equations given above, and let us denote them as $r_1 = s_1$, $r_2 = s_2$, and $r_3 = s_3$, respectively. Let u be the context $g(\square)$. Then (u, E_G) is a subterm permutation group, and any term t that is an instance of $g(f(v_1, f(v_2, f(v_3, v_4))))$ is in the domain of (u, E_G). In particular, the term $t \equiv g(f(a, f(b, f(c,d))))$ is in the domain of (u, E_G). From E_G we obtain the mappings

$$\phi_{r_1,s_1} : g(f(v_1, f(v_2, f(v_3, v_4)))) \mapsto g(f(v_2, f(v_1, f(v_3, v_4))))$$

$$\phi_{r_2,s_2} : g(f(v_1, f(v_2, f(v_3, v_4)))) \mapsto g(f(v_1, f(v_3, f(v_2, v_4))))$$

$$\phi_{r_3,s_3} : g(f(v_1, f(v_2, f(v_3, v_4)))) \mapsto g(f(v_1, f(v_2, f(v_4, v_3))))$$

Thus $\phi_{r_1,s_1}(g(f(a, f(b, f(c,d))))) \equiv g(f(b, f(a, f(c,d))))$, $\phi_{r_2,s_2}(g(f(a, f(b, f(c,d))))) \equiv g(f(a, f(c,\ f(b,\ d))))$, and $\phi_{r_3,s_3}(g(f(a, f(b,\ f(c,d))))) \equiv g(f(a, f(b,\ f(d,c))))$. Note that these terms are again in the domain of (u, E_G), so we can obtain by composition the mapping

$$\phi_{r_2,s_2} \cdot \phi_{r_1,s_1} : g(f(v_1, f(v_2, f(v_3, v_4)))) \mapsto g(f(v_2, f(v_3, f(v_1, v_4))))$$

and also other compositions, such as $\phi_{r_2,s_2} \cdot \phi_{r_1,s_1}$. We also obtain inverses, and the group $\Phi^*(G)$ is then the set of all mappings of the form

$$\phi : g(f(v_1, f(v_2, f(v_3, v_4)))) \mapsto g(f(w_1, f(w_2, f(w_3, w_4))))$$

where the w_i are a permutation of the v_i.

2.1 Stratified Subterm Permutation Groups

We would now like to consider terms t having more than one subterm permutation group, where these groups may be applied simultaneously at different positions of t. We construct a group structure which includes all of these subterm permutation groups together. We want to restrict these groups, however, so that they do not overlap, that is, the action of one of these groups does not destroy the term structure of any other group. For this, we need a finite set u_i of contexts indicating the positions where each subterm permutation group applies. We therefore define a *labeled term* $(t, u_1, \ldots, u_n)$ to be a term t together with a sequence $u_1, \ldots, u_n$ of 1-contexts having t as a context instance. Each context u_i can be thought of as a position of t, namely, the position of $\square$ in u_i. The subterms of t at the positions $u_1, \ldots, u_n$ can be thought of as labeled with the integers 1 through n, respectively. Sometimes we just refer to t as a labeled term, with the u_i not explicitly mentioned. A *stratified permutation group* G is a (possibly empty) sequence $E_1, E_2, \ldots, E_n$ of uniform sets of leaf permutative equations. Each E_i must be uniform, but there may be equations $r_i = s_i$ in E_i and $r_j = s_j$ in E_j such that r_i and

r_j are not variants. Thus, $E_i \cup E_j$ need not be uniform. As before, we obtain mappings from the E_i, and the contexts u_i specify where the equations E_i may be applied. A labeled term $(t, u_1, \ldots, u_n)$ is in the domain of G if for every i, t is in the domain of the subterm permutation group (u_i, E_i), that is, for some Θ, $t \equiv u_i[r_i\Theta]$, where r_i is a redex for E_i. Also, we require that t satisfy an *independence condition*, namely, if $u_j > u_i$, that is, there exists a term w_{ij} such that $u_j \equiv u_i[w_{ij}]$, then there is a Θ such that $u_j \equiv u_i[r_i\Theta]$. Thus the labeled positions must be chosen so that the various redexes do not overlap. We also require that for all i and for every equation $r = s$ in E_i, $E \models r = s$. This insures that $S((t, u_1, \ldots, u_n), G)$ is a subset of the E-equivalence class of t. We can consider the stratified permutation group $G = (E_1, \ldots, E_n)$ and the labeled term $(t, u_1, \ldots, u_n)$ as obtained from the (non-stratified groups) $G_i = (u_i, E_i)$ and the term t; thus, in a sense the pair $((t, u_1, \ldots, u_n), G)$ is obtained from the groups G_i and the term t. We will informally speak of the group G and the labeling $(t, u_1, \ldots, u_n)$ as constructed from the groups G_i.

Recall that $\Phi(u_i, E_i)$ is a set of generators of the subterm permutation group (u_i, E_i). Then each ϕ_{r,s,u_i} in $\Phi(u_i, E_i)$ maps terms $u_i[r\Theta]$ to $u_i[s\Theta]$, and terms not of the form $u_i[r\Theta]$ for some Θ map to themselves. We want to extend ϕ_{r,s,u_i} to a mapping of labeled terms, by mapping the "positions" u_i in the obvious manner. We extend ϕ_{r,s,u_i} to a mapping $\phi^i_{r,s}$ from labeled terms $(t, u_1, \ldots, u_n)$ in the domain of G to labeled terms $(t', u'_1, \ldots, u'_n)$ in the following way:

$$\phi^i_{r,s}((t, u_1, \ldots, u_n)) = (\phi_{r,s,u_i}(t), \phi_{r,s,u_i}(u_1), \ldots, \phi_{r,s,u_i}(u_n))$$

The idea is to keep track of how the permutation ϕ_{r,s,u_i} maps the labeled subterms of t, and to put the labels in their proper places in $\phi_{r,s,u_i}(t)$.

We now show that the mappings $\phi^i_{r,s}$ map terms in the domain of G to terms in the domain of G and thus can be composed, forming a group structure under function composition.

Theorem 2.3 *Suppose $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group and $(t, u_1, \ldots, u_n)$ is a labeled term in the domain of G. Let $r_i = s_i$ be an equation in E_i. Then $\phi^i_{r_i,s_i}((t, u_1, \ldots, u_n))$ is also in the domain of G.*

Proof. Let $(t', u'_1, \ldots, u'_n)$ be $\phi^i_{r_i,s_i}((t, u_1, \ldots, u_n))$. Let r_j be a redex for E_j, $1 \leq j \leq n$. We show that $(t', u'_1, \ldots, u'_n)$ is also in the domain of G. First we show that for all j, there is a Θ_j such that $t' \equiv u'_j[r_j\Theta_j]$. This involves three cases, $u_i < u_j$, $u_i > u_j$, and u_i and u_j are independent.

Suppose $u_i < u_j$. Then there is a Θ_i such that $u_j \equiv u_i[r_i\Theta_i]$ and there is a Θ_j such that $t \equiv u_j[r_j\Theta_j]$ since $(t, u_1, \ldots, u_n)$ is in the domain of G. Thus $t \equiv u_i[r_i\Theta_i][r_j\Theta_j]$. Then $t' \equiv u_i[s_i\Theta_i][r_j\Theta_j]$ and $u'_j \equiv u_i[s_i\Theta_i]$. Hence $t' \equiv u'_j[r_j\Theta_j]$.

Suppose $u_i > u_j$. Then there is a Θ_j such that $u_i \equiv u_j[r_j\Theta_j]$ and there is a Θ_i such that $t \equiv u_i[r_i\Theta_i]$, since $(t, u_1, \ldots, u_n)$ is in the domain of G. Hence $t \equiv u_j[r_j\Theta_j][r_i\Theta_i]$. Then $t' \equiv u_j[r_j\Theta_j][s_i\Theta_i]$. Also, $u'_j \equiv u_j$. Thus $t' \equiv u'_j[r_j\Theta_j][s_i\Theta_i]$.

Suppose u_i and u_j are independent. Let u_{ij} be a context such that $t \equiv u_{ij}[r_i\Theta_i, r_j\Theta_j]$ and $u_i \equiv u_{ij}[\square, r_j\Theta_j]$ and $u_j \equiv u_{ij}[r_i\Theta_i, \square]$. Then $u'_j \equiv u_{ij}[s_i\Theta_i, \square]$ and $t' \equiv u_{ij}[s_i\Theta_i, r_j\Theta_j]$. Hence $t' \equiv u'_j[r_j\Theta_j]$.

We also need to check the independence conditions, that is, if $u'_k > u'_j$ then there is a Θ_j such that $u'_k \equiv u'_j[r_j\Theta_j]$. With a little work, we can show that $u'_k > u'_j$ iff $u_k > u_j$. That is, the mappings $\phi^i_{r_i,s_i}$ do not change the relative order of the contexts u_j. Therefore, we only need to consider the case $u_k > u_j$ and check that there is a Θ_j such that $u'_k \equiv u'_j[r_j\Theta_j]$.

Suppose then that $u_i < u_j < u_k$. Then there is a Θ_i such that $u_j \equiv u_i[r_i\Theta_i]$ and there is a Θ_j such that $u_k \equiv u_j[r_j\Theta_j]$. Also, there is a Θ_k such that $t \equiv u_k[r_k\Theta_k]$. Therefore $u_k \equiv u_i[r_i\Theta_i][r_j\Theta_j]$. Also, $t \equiv u_i[r_i\Theta_i][r_j\Theta_j][r_k\Theta_k]$. Then we have that $t' \equiv u_i[s_i\Theta_i][r_j\Theta_j][r_k\Theta_k]$, $u'_j \equiv u_i[s_i\Theta_i]$, and $u'_k \equiv u_i[s_i\Theta_i][r_j\Theta_j]$. Therefore $u'_k \equiv u'_j[r_j\Theta_j]$ as required.

Suppose that $u_i > u_j$ and $u_j < u_k$. Then we have to consider three cases corresponding to the three possible ordering relationships between u_i and u_k. We omit the arguments, since they are similar to others given here.

Suppose finally that u_i and u_j are independent and that $u_j < u_k$. Let u_{ij} be as before. Then $t \equiv u_k[r_k\Theta_k]$ and $u_k \equiv u_j[r_j\Theta_j]$ and $u_j \equiv u_{ij}[r_i\Theta_i, \square]$. It follows that $t \equiv u_{ij}[r_i\Theta_i, r_j\Theta_j[r_k\Theta_k]]$. We want u_i to be the context that has a $\square$ in the $r_i\Theta_i$ position of t, so $u_i \equiv u_{ij}[\square, r_j\Theta_j[r_k\Theta_k]]$. Note also that $u_k \equiv u_{ij}[r_i\Theta_i, r_j\Theta_j]$. Then $t' \equiv u_{ij}[s_i\Theta_i, r_j\Theta_j[r_k\Theta_k]]$, $u'_j \equiv u_{ij}[s_i\Theta_i, \square]$, and $u'_k \equiv u_{ij}[s_i\Theta_i, r_j\Theta_j]$. Therefore $u'_k \equiv u'_j[r_j\Theta_j]$ as required by the independence condition. $\square$

Let $\Phi_i(E_i)$ be $\{\phi^i_{r,s} : \phi_{r,s,u_i} \in \Phi(u_i, E_i)\}$. Let $\Phi(G)$ be $\Phi_1(E_1) \cup \cdots \cup \Phi_n(E_n)$. We call $\Phi(G)$ as before the *generators* of the stratified permutation group G. We then obtain a group of mappings generated by $\Phi(G)$ under the operation of function composition. We call this group $\Phi^*(G)$. For computational purposes, it is more convenient to represent this group of mappings by $\Phi(G)$ rather than by $\Phi^*(G)$. Let $\Phi^*_i(E_i)$ be the group generated by $\Phi_i(E_i)$ under the operation of function composition. It turns out that the group $\Phi^*(G)$ is a direct product of the groups $\Phi^*_i(E_i)$, as we will show later. This is not obvious, since the mappings interact with each other. In particular, some mappings $\phi^i_{r,s}$ change the positions u_j at which other mappings $\phi^j_{r',s'}$ apply. As before, $S((t, u_1, \ldots, u_n), G) = \{\phi((t, u_1, \ldots, u_n)) : \phi \in \Phi^*(G)\}$. We sometimes abbreviate $(t, u_1, \ldots, u_n)$ as t, with the labels u_i being implicit. Also, assuming that $E \models r = s$ for all $\phi_{r,s}$ in $\Phi(u_i, E_i)$, we can show that for all $t' \in S(t, G)$, $E \models t = t'$ (ignoring the labels). However, there may be terms t' that are not members of $S(t, G)$ such that $E \models t = t'$. Furthermore, $S(t, G)$ is finite, since all of its elements have the same length (number of symbol occurrences) as t. Let $\overline{u}$ be a notation for $(u_1, u_2, \ldots, u_n)$. We often write $(t, u_1, \ldots, u_n)$ as $(t, \overline{u})$. If $(t', \overline{u}') \in S((t, \overline{u}), G)$, then $S((t', \overline{u}'), G) = S((t, \overline{u}), G)$, since the functions $\Phi^*(G)$ are closed under the group operation (function composition). We are sometimes not interested in the labels, so we define $S'((t, \overline{u}), G)$ as $\{t' : (\exists \overline{u}')(t', \overline{u}) \in S((t, \overline{u}), G)\}$, that is, $S'((t, \overline{u}), G)$ consists of the elements of $S((t, \overline{u}), G)$ with the labels removed.

One can easily construct subterm permutation groups G, but finding a labeled term in the domain of G is nontrivial – more precisely, given a t, finding u_i (if they exist) so that $(t, u_1, \ldots, u_n)$ is in the domain of G is not trivial, though it can be done by exhaustive search. Thus we typically work with pairs $((t, \overline{u}), G)$, where $(t, \overline{u})$ is a labeled term in the domain of G, rather than working with stratified permutation groups G alone. We call such a pair $((t, \overline{u}), G)$, a *pointed group.*

Example 2.4 We now give an example of a stratified subterm permutation group. Let G be (E_1, E_2), where E_1 is the single equation $\{f(x,y) = f(y,x)\}$ and E_2 is the single equation $\{g(x,y) = g(y,x)\}$. Let t be the labeled term $(f(a, g(b,c)), \square, f(a, \square))$. Then t is in the domain of G. In fact, t is a *transitional* term for G, since one of the contexts is $\square$, as we shall see later when "transitional" is defined. The first context, $\square$, signifies that the equations E_1 apply at the top of t, and the second context, $f(a, \square)$, indicates that the equations E_2 apply to the second argument of f. We give some mappings. The mapping $\phi^1_{f(x,y),f(y,x)}$ applied to t yields $(f(g(b,c),a), \square, f(\square, a))$. Note that the second context has changed. The new context $f(\square, a)$ indicates that E_2 now applies to the first argument of f. Thus it is possible to apply some ϕ for E_2 to this term if desired; that is, the resulting term is in the domain of the (non-stratified) subterm permutation group $(f(\square, a), E_2)$. Let us now consider the original labeled term t again. The mapping $\phi^2_{g(x,y),g(y,x)}$ applied to t yields $(f(a, g(c,b)), \square, f(a, \square))$. Note that none of the contexts have changed, since the context $f(a, \square)$ for E_2 has its occurrence of $\square$ "inside" the occurrence of $\square$ in the context $\square$ for E_1. Also, $S(t, G) = \{(f(a, g(b,c)), \square, f(a, \square))$, $(f(a, g(c,b)), \square, f(a, \square))$, $(f(g(b,c),a), \square, f(\square, a))$, $(f(g(c,\ b),\ a), \square, f(\square, a))\}$.

Example 2.5 We now show how the conditions on non-overlapping of the various subterm permutation groups can be violated. Suppose t is $f(x, f(y, f(z,w)))$ and u_1 is $\square$ and u_2 is $f(x, \square)$. Suppose $E_1 = E_2 = \{f(x, f(y,z)) = f(y, f(x,z)),\ f(x, f(y,z)) = f(x, f(z,y))\}$. Then the non-overlapping condition is violated, because the action of E_1 can destroy the term structure for E_2. That is, applying the equation $f(x, f(y,z)) = f(x, f(z,y))$ as specified by u_1, that is, at the top level of t, we obtain the term $f(x, f(f(z,w),y))$ from t; the subterm $f(y, f(z,w))$ of t needed by E_2 has been changed, so that the equations in E_2 can no longer apply at the position specified by u_2. For this reason, we impose the condition that the terms correspoding to the various subterm permutation groups cannot overlap at non-variable positions, which excludes this example from being a stratified subterm permutation group.

2.2 Marked Terms

There is another approach to the definition of stratified subterm permutation groups that is in some ways simpler. The idea is to specify the positions α_i in t at which the equations E_i apply by "marking" the function symbols at positions α_i in t, that is, replacing them by new function symbols that appear nowhere else in t. It is possible that the same function symbol f may appear in more than one location in t and be marked in two different ways. Other occurrences of f in t may be left unmarked. We also need to mark the equations E_i so that the marked term t will still have a redex for E_i. The technique of using marked terms considerably simplifies the formalism. Let the symbols $f_1^m, f_2^m, f_3^m, \cdots$ be a fixed alphabet of *marked* function symbols that do not appear in unmarked terms. A *marked term* t is a term that may contain marked function symbols as well as unmarked function symbols, but each marked function symbol has at most one occurrence in t. A *marked* equation is an equation $r = s$ where the topmost function symbols in r and s are marked and identical. A *marked* stratified subterm permutation group G' is a sequence $((E_1', g_1), (E_2', g_2), \ldots, (E_n', g_n))$ where each E_i' is a uniform set of marked equations and g_i is an unmarked function symbol. Note that all the equations in E_i' must be marked with the same marked function symbol (since E_i' is uniform). The g_i are included to recall what the top-level symbol of the E_i' was before it was marked. The symbols g_i need not all be distinct. However, we require that if i and j are distinct, then the (marked) top-level function symbols of E_i' and E_j' are distinct. A marked term t' is in the domain of G' if for all i, t' contains a redex for E_i', that is, a subterm that is an instance of the left-hand side of an equation in E_i'. As before, we call a pair (t', G') where t' is a labeled term in the domain of G', a *pointed (marked) group*. We note that if the equation $r = s$ is in E_i' and there is a substitution Θ such that $t' \equiv t'[r\Theta]$, then $t'[s\Theta]$ is also a marked term in the domain of G'.

We now specify correspondences between labeled groups and marked groups. Suppose $G = (E_1, E_2, \ldots, E_n)$ is a stratified subterm permutation group, and the labeled term $(t, u_1, \ldots, u_n)$ is in the domain of G. Suppose α_i are the positions of $\Box$ in u_i, respectively. Let $\overline{u}$ be a notation for $(u_1, u_2, \ldots, u_n)$. We often write $(t, u_1, \ldots, u_n)$ as $(t, \overline{u})$. We define the function m from labeled terms to marked terms by $m((t, \overline{u})) \equiv t'$, where t' is the term t with the function symbol f at position α_i replaced by f_i^m. We also define the function m from labeled groups to marked groups by $m(G) = ((E_1', g_1), (E_2', g_2), \ldots, (E_n', g_n))$, where E_i' are the equations in E_i with their topmost function symbol occurrences replaced by f_i^m, and g_i is the top-level function symbol of the equations E_i. Thus if an equation $g_i(r_1, \cdots, r_k) = g_i(s_1, \cdots, s_k)$ is in E_i, then the equation $f_i^m(r_1, \cdots, r_k) = f_i^m(s_1, \cdots, s_k)$ is in E_i'. Note that the f_i^m are all distinct, but the g_i need not all be distinct. There is a one-to-one correspondence between pointed labeled groups and (a subset of the) pointed marked groups, because the extra symbols g_i give enough information to reconstruct the pointed labeled group $((t, \overline{u}), G)$ from the marked term/group $(m((t, \overline{u})), m(G))$. We define the inverse function um (unmark) by $um : (m((t, \overline{u})), m(G)) \mapsto (t, G)$. We also define this function um on groups by $um(m(G)) = G$. However, the marked term $m((t, \overline{u}))$ does not have enough information to reconstruct t, since the original symbols g_i have been lost. Therefore, we define $um^{G'}(t')$ to be the first component of $um(t', G')$. Thus $um^{G'}$ removes the markings from a term; in other words, the top-level symbol f_i^m of E_i' is replaced again by g_i. Also, the contexts u_i are reconstructed. It follows that $um^{m(G)}(m((t, \overline{u}))) = (t, \overline{u})$ if $(t, \overline{u})$ is in the domain of G.

We now define $S(t', G')$ where t' is a marked term and $G' = ((E_1', g_1), (E_2', g_2), \ldots, (E_n', g_n))$. Given a set E' of marked equations, let $\Phi(E')$ be the set of mappings $\phi_{r,s}$ for equations $r = s$ in E', where $\phi_{r,s}$ is defined as before by $\phi_{r,s} : u[r\Theta] \mapsto u[s\Theta]$, for all contexts u and all substitutions Θ. Thus $w \rightarrow_{E'} \phi_{r,s}(w)$ if w has a redex for E'. Let $\Phi(G')$ be $\Phi(E'_1) \cup \cdots \cup \Phi(E'_n)$. We call $\Phi(G')$ the *generators* of G', as before. Let $\Phi^*(G)$ be the group generated by $\Phi(G')$. Finally, $S(t', G') = \{\phi(t') : \phi \in \Phi^*(G')\}$. As before, if $w \in S(t', G')$, then $S(w, G') = S(t', G')$. We now give an alternative definition which is interesting because it is stated purely in terms of the equational theory; that is, it is a semantic definition, not a syntactic one. The equational theory $E'_1 \cup \cdots \cup E'_n$ here considered consists of *marked* equations, so it is not the same as E and not even the same as $E_1 \cup \cdots \cup E_n$. Furthermore, we are considering equivalence classes of marked terms, so the result does not directly transfer to equivalence classes of unmarked terms.

Theorem 2.6 *Suppose t' is a marked term in the domain of a marked stratified subterm permutation group $G' = ((E'_1, g_1), \cdots, (E'_n, g_n))$. Then $S(t', G') = \{w : E'_1 \cup \cdots \cup E'_n \models t' = w\}$.*

Proof. Let E' be $E'_1 \cup \cdots \cup E'_n$. Suppose $E' \models t' = w$. Then by Birkhoff's theorem, $t' \leftrightarrow^*_{E'} w$. We want to show that $w \in S(t', G')$. By the definition of $\Phi(G')$, if $v_1 \rightarrow_{E'} v_2$ then there exists $\phi \in \Phi(G')$ such that $v_2 \equiv \phi(v_1)$. Since groups contain inverses, there exists $\phi \in \Phi^*(G')$ such that $v_1 \equiv \phi(v_2)$; thus, if $v_1 \leftrightarrow_{E'} v_2$ then there exists $\phi \in \Phi(G')$ such that $v_2 \equiv \phi(v_1)$. Since $\Phi^*(G')$ is closed under composition of ϕ, we have that if $v_1 \leftrightarrow^*_{E'} v_2$ then there exists $\phi \in \Phi(G')$ such that $v_2 \equiv \phi(v_1)$. Since $t' \leftrightarrow^*_{E'} w$, there exists $\phi \in \Phi^*(G')$ such that $\phi(t') \equiv w$. By definition of $S(t', G')$, $w \in S(t', G')$. The other direction is similar. □

The property that makes possible this agreement between E'-equivalence classes and subterm permutation groups is that the E'_i are non-overlapping, which we formalize in the following definition.

Definition 2.7 *A set $\{E_1, E_2, \cdots, E_n\}$ of uniform sets of equations is* orthogonal *if no left-hand side of an equation in E'_j can unify with a non-variable subterm of an equation in E'_i, for $i \neq j$, and if no left-hand side of an equation in E'_i unifies with a proper non-variable subterm of any other left-hand side of an equation in E'_i, for any i.*

Theorem 2.8 *Suppose $\{E_1, E_2, \cdots, E_n\}$ is orthogonal. Then for every term t there is a labeling $\overline{u}$ and a stratified subterm permutation group G such that $S'((t, \overline{u}), G)$ $= \{t' : E_1 \cup E_2 \cup \cdots \cup E_n \models t = t'\}$.*

Proof. Let $\{u_1, u_2, \ldots, u_m\}$ be the set of contexts u such that there exists a substitution Θ and an i and an E_i-redex r such that $t \equiv u[r\Theta]$. Then these contexts u_i satisfy the independence condition of section 2.1 for stratified subterm permutation groups. Let $\overline{u}$ be $(u_1, u_2, \ldots, u_m)$, and let $(E'_1, E'_2, \ldots, E'_m)$ be a listing of the E_i such that if $t \equiv u_j[r\Theta]$ and r is an E_k-redex, then E'_j is E_k. It follows that $(t, \overline{u})$ is in the domain of the stratified subterm permutation group $G = (E'_1, E'_2, \ldots, E'_m)$. Furthermore, reasoning as theorem 2.6, the conclusion of the theorem holds. □

We note that this result deals with unmarked terms and equations, in contrast to theorem 2.6. Returning to the relationship between marked and unmarked subterm permutation groups, we now define $S((t,\overline{u}),G)$ in terms of $S(m((t,\overline{u})),m(G))$.

Lemma 2.9 *Suppose $G = (E_1, \cdots, E_n)$ is a labeled group. Suppose $G' = ((E_1', g_1), \cdots, (E_n', g_n))$ is the corresponding marked subterm permutation group, that is, $G' = m(G)$, and w is in the domain of G'. Let $r_i = s_i$ be an equation in E_i, and let $r_i' = s_i'$ be the corresponding equation in E_i'. Then*

$$um(\phi_{r_i', s_i'}(w)) \equiv \phi^i_{r_i, s_i}(um(w))$$

.

Proof. Let $(t, u_1, \ldots, u_n)$ be $um(\phi_{r_i', s_i'}(w))$ and let $(t', u_1', \ldots, u_n')$ be $\phi^i_{r_i, s_i}(um\ (w))$. We want to show that $t \equiv t'$ and for all j, $u_j \equiv u_j'$. Let α_i be the position of f_i^m in u_i. Both t and t' have an instance $r\Theta$ of r at position α_i replaced by $s\Theta$, since all marks were removed. It remains to consider each f_j^m. We note that $u_i \equiv u_i'$ since both are unchanged by the mappings; both end up with a $\Box$ in position α_i. We then consider the other u_j. There are three cases – f_j^m appears at a position below f_i^m, f_j^m appears at a position above f_i^m, and they appear at independent positions. Each such f_j^m becomes an occurrence of $\Box$ in u_j and u_j' by the unmarking operations, and we need to check that these occurrences are in the same position. In the first case, the marked function symbol is mapped the same way by the two operations; in the second case, the positions do not change; and in the third case, the positions do not change, but the rest of the context is rewritten in the same way. By fairly straightforward arguments, the details can be filled in and the result follows. $\Box$

Theorem 2.10 *Suppose that G is a stratified subterm permutation group (using labels) and $(t, u_1, \cdots, u_n)$ is a labeled term in the domain of G. Suppose that $m(G)$ is the corresponding marked stratified subterm permutation group and $m((t,\overline{u})$ is a labeled term in the domain of $m(G)$. Then $S((t, u_1, \cdots, u_n), G) = \{um^{m(G)}(w) : w \in S(m((t,\overline{u})), m(G))\}$.*

Proof. By repeated application of the lemma, using the fact that the $\phi^i_{r_i,s_i}$ generate $\Phi^*(G)$ and the $\phi_{r_i',s_i'}$ generate $\Phi^*(G')$. $\Box$

Example 2.11 We now give an example of a stratified subterm permutation group using marked terms, corresponding to example 2.4. Let G' be $((E_1', g_1), (E_2', g_2))$, where E_1' is the single equation $\{f_1^m(x,y) = f_1^m(y,x)\}$ and E_2' is the single equation $\{f_2^m(x,y) = f_2^m(y,x)\}$ and the symbols g_i are not relevant here. Let t' be the marked term $f_1^m(a, f_2^m(b,c))$. Then t' is in the domain of G'. Note that the equations E_1' apply at the top of t', and the equations E_2' apply to one of the proper subterms of t'. We give some terms in $S(t', G')$. The term $f_1^m(f_2^m(b,c),a)$ is in $S(t', G')$, since $E_1' \cup E_2' \models f_1^m(a, f_2^m(b,c)) = f_1^m(f_2^m(b,c),a)$. Note that the term $f_1^m(f_2^m(b,c),a)$ is also in the domain of G'. Finally, $S(t', G') = \{f_1^m(a, f_2^m(b,c)),\ f_1^m(a, f_2^m(c,b)),\ f_1^m(f_2^m(b,c),a),\ f_1^m(f_2^m(c,\ b),\ a)\}$.

2.3 Properties of Subterm Permutation Groups

We first consider commutativity properties of stratified subterm permutation groups.

Definition 2.12 *A* term transposition *is an equation of the form $t[x,y] = t[y,x]$, where t is a 2-context. If the equation $r = s$ is a term transposition, then we also call $\phi_{r,s}$ and $\phi^i_{r,s}$ term transpositions.*

We note that if $r = s$ is a leaf permutation, then $\phi_{r,s}$ can be expressed as a composition of term transpositions, that is, a composition of mappings $\phi_{r',s'}$, where the equations $r' = s'$ are term transpositions.

Theorem 2.13 *Suppose $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group and $r_i = s_i \in E_i$ and $r_j = s_j \in E_j$ for $i \neq j$. Then for any labeled term t in the domain of G, $\phi^i_{r_i,s_i}(\phi^j_{r_j,s_j}(t)) = \phi^j_{r_j,s_j}(\phi^i_{r_i,s_i}(t))$.*

Proof. Let us express the labeled term t more fully as $(t, u_1, \ldots, u_n)$, where the u_k are 1-contexts. We recall that in general, there exists $\overline{u}'$ such that $\phi^j_{r,s}(t, \overline{u}) = (\phi_{r,s}(t), \overline{u}')$. We first consider how the unlabeled term t maps, and then briefly comment on the mapping of the contexts $\overline{u}$. Since $\phi^i_{r_i,s_i}$ and $\phi^j_{r_j,s_j}$ can be expressed as products of term transpositions, it suffices to prove the theorem for the case when both $\phi^i_{r_i,s_i}$ and $\phi^j_{r_j,s_j}$ are term transpositions. Let us write the equation $r_i = s_i$ as $t_i[x,y] = t_i[y,x]$ and the equation $r_j = s_j$ as $t_j[x,y] = t_j[y,x]$, where t_i and t_j are 2-contexts. Suppose that u_i and u_j are independent. Then there is a 2-context u_{ij} and terms v_i such that t is $u_{ij}[t_i[v_1, v_2], t_j[v_3, v_4]]$. In this case, $\phi_{r_i,s_i}(\phi_{r_j,s_j}(t)) \equiv u_{ij}[t_i[v_2, v_1], t_j[v_4, v_3]]$ and $\phi_{r_j,s_j}(\phi_{r_i,s_i}(t)) \equiv u_{ij}[t_i[v_2, v_1], t_j[v_4, v_3]]$ also, since both term transpositions act on independent positions. Suppose that one of u_i and u_j is a context instance of the other; without loss of generality, assume that u_j is a context instance of u_i. Thus t can be written as $u_i[t_i[w[t_j[v_1, v_2]], v_3]]$ where w is a 1-context. That is to say, the occurrence of $\square$ in u_j is "within" the occurrence of x in $u_i[t_i[x,y]]$. Then $\phi_{r_j,s_j}(t) \equiv u_i[t_i[w[t_j[v_2, v_1]], v_3]]$ and $\phi_{r_i,s_i}(\phi_{r_j,s_j}(t)) \equiv u_i[t_i[v_3, w[t_j[v_2, v_1]]]]$. Now, $\phi_{r_i,s_i}(t) \equiv u_i[t_i[v_3, w[t_j[v_1, v_2]]]]$ and $\phi_{r_j,s_j}(\phi_{r_i,s_i}(t)) \equiv u_i[t_i[v_3, w[t_j[v_2, v_1]]]]$. These two terms are the same. The idea is that since the redexes are independent and the rules are linear, they can be done in either order.

We also need to check that the contexts u_k are mapped the same way by $\phi^i_{r_i,s_i}(\phi^j_{r_j,s_j}(t, \overline{u}))$ and $\phi^j_{r_j,s_j}(\phi^i_{r_i,s_i}(t, \overline{u}))$. This is somewhat laborious, but follows the same general outline as the argument for t just given. Let $(t', u'_1, \ldots, u'_n)$ be $\phi^i_{r_i,s_i}(\phi^j_{r_j,s_j}(t, \overline{u}))$, and let $(t'', u''_1, \ldots, u''_n)$ be $\phi^j_{r_j,s_j}(\phi^i_{r_i,s_i}(t, \overline{u}))$. We just showed that $t' \equiv t''$. We want to show that for all k, $u'_k \equiv u''_k$. Recall that $u'_k \equiv \phi^i_{r_i,s_i}(\phi^j_{r_j,s_j}(u_k))$ and $u''_k \equiv \phi^j_{r_j,s_j}(\phi^i_{r_i,s_i}(u_k))$. Thus $u'_k \equiv \phi_{r_i,s_i,\phi_{r_j,s_j,u_j}(u_i)}\ (\phi_{r_j,s_j,u_j}\ (u_k))$ and $u''_k \equiv \phi_{r_j,s_j,\phi_{r_i,s_i,u_i}(u_j)}\ (\phi_{r_i,s_i,u_i}\ (u_k))$. For u_k such that $u_i \geq u_k$ and $u_j \geq u_k$, $u'_k \equiv u''_k \equiv u_k$, since such contexts are unchanged by these mappings ϕ. For u_k such that neither $u_i \geq u_k$ nor $u_j \geq u_k$, $u'_k \equiv \phi_{r_i,s_i}(\phi_{r_j,s_j}(u_k))$ and $u''_k \equiv \phi_{r_j,s_j}(\phi_{r_i,s_i}(u_k))$. These are equal, reasoning exactly as above for t' and t'', since u_k can be considered as just another term. The only case that remains is when $u_i \geq u_k$ but not $u_j \geq u_k$, or when $u_j \geq u_k$ but not $u_i \geq u_k$. If $u_i \geq u_k$ but not $u_j \geq u_k$, then $u'_k \equiv u''_k \equiv \phi_{r_j,s_j}(u_k)$, and if $u_j \geq u_k$ but not $u_i \geq u_k$, then $u'_k \equiv u''_k \equiv \phi_{r_i,s_i}(u_k)$. Of course, using marked terms, this argument about the contexts is immediate. This completes the proof. $\square$

Corollary 2.14 *Suppose $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group and $\phi \in \Phi^*(G)$. Then there are functions $\phi_i \in \Phi_i^*(E_i)$ such that for any labeled term t in the domain of G, $\phi(t) = \phi_1 \cdot \phi_2 \cdot \ldots \cdot \phi_n(t)$.*

Proof. By the theorem, mappings from different E_i commute with one another, so we can always interchange them until all the mappings from a given set E_i of equations are adjacent to one another. $\square$

We now develop some results to show that $\Phi^*(G)$ is a direct product of the groups $\Phi_i^*(E_i)$.

Definition 2.15 *The* frontier *$fr(t)$ of a term t is the set of variable and constant symbols that occur in t. Thus, the frontier of $f(x, f(y, x))$ is $\{x, y\}$.*

Lemma 2.16 *Suppose $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group and $(t, \overline{u})$ is in the domain of G. Then for all terms t' in $S'((t, \overline{u}), G)$, $fr(t') = fr(t)$.*

Proof. It suffices to show that for all i and all equations $r = s$ in E_i and all terms v, $fr(\phi_{r,s}(v)) = fr(v)$. This follows because $\phi_{r,s}$ permutes the terms appearing at certain positions in v, or else $\phi_{r,s}(v) \equiv v$. $\square$

Lemma 2.17 *Let t be a term having at most one occurrence of each subterm. Then for distinct subterms v_1 and v_2 of t, $fr(v_1) \cap fr(v_2) = \{\}$. Furthermore, if $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group and ϕ and ϕ' are distinct elements of $\Phi^*(G)$, and $(t, u_1, \ldots, u_n)$ is a term in the domain of G, then $\phi(t) \not\equiv \phi'(t)$ (ignoring the contexts).*

Proof. The part about $fr(v_1) \cap fr(v_2)$ follows since t has at most one occurrence of each variable and constant symbol. Suppose ϕ and ϕ' are as in the lemma. Let us express ϕ as $\phi_1 \cdot \phi_2 \cdot \ldots \cdot \phi_n$ where the functions ϕ_i are in $\Phi_i^*(E_i)$. Let us express ϕ' as $\phi'_1 \cdot \phi'_2 \cdot \ldots \cdot \phi'_n$ where the functions ϕ'_i are in $\Phi_i^*(E_i)$. Let u_i be a minimal context (in the ordering on contexts) from the set $\{u_1, \ldots, u_n\}$ such that $\phi_i(t) \not\equiv \phi'_i(t)$. Ignoring the labels, we can express t as $u_i[r\Theta]$ where r is a redex for E_i. Let us write r as $r[x_1, \ldots, x_m]$ where all variables are listed. Then $\phi_i(t) \equiv u_i[r[y_1\Theta, \ldots, y_m\Theta]]$ and $\phi'_i(t) \equiv u_i[r[z_1\Theta, \ldots, z_m\Theta]]$ where the y_i and z_i are permutations of the x_i, and for some

j, $y_j \not\equiv z_j$. It follows that $y_j\Theta \not\equiv z_j\Theta$, and in fact, $fr(y_j\Theta)$ and $fr(z_j\Theta)$ are disjoint. Let γ be the position of the $\Box$ in u_i. Then $\phi(t)|_\gamma \equiv r[v_1, \ldots, v_m]$ and $\phi'(t)|_\gamma \equiv r[v'_1, \ldots, v'_m]$ where $fr(v_k) = fr(y_k\Theta)$ for all k and $fr(v'_k) = fr(z_k\Theta)$ for all k. This implies that $fr(v_j) \neq fr(v'_j)$, so $\phi(t)|_\gamma \not\equiv \phi'(t)|_\gamma$, and thus $\phi(t) \not\equiv \phi'(t)$ (ignoring the labels). $\Box$

Theorem 2.18 *Suppose $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group. Then $\Phi^*(G)$ is a direct product of the groups $\Phi^*_i(E_i)$.*

Proof. By the preceding results, if ϕ_i and ϕ'_i are elements of $\Phi^*_i(E_i)$, then $\phi_1 \cdot \phi_2 \cdot \ldots \cdot \phi_n \cdot \phi'_1 \cdot \phi'_2 \cdot \ldots \cdot \phi'_n = (\phi_1 \cdot \phi'_1) \cdot \ldots \cdot (\phi_n \cdot \phi'_n)$. Therefore $\Phi^*(G)$ is a homomorphic image of the direct product, hence is a quotient group of this direct product. To show that it is a direct product, we need to show that if ϕ_i and ϕ'_i are distinct elements of $\Phi^*_i(E_i)$, then $\phi_1 \cdot \phi_2 \cdot \ldots \cdot \phi_n$ and $\phi'_1 \cdot \phi'_2 \cdot \ldots \cdot \phi'_n$ are distinct, that is, there exists a term t such that $\phi_1 \cdot \phi_2 \cdot \ldots \cdot \phi_n(t)$ and $\phi'_1 \cdot \phi'_2 \cdot \ldots \cdot \phi'_n(t)$ are distinct. This can be shown by letting t be a term with at most one occurrence of each subterm, using lemma 2.17. $\Box$

We now consider the effect of subterm permutation groups on subterms of t. If $f(t_1, \ldots, t_n)$ is a term then we call t_i the *i-th subterm* (or the i-th top-level subterm) of t. We also denote this by $t|_i$.

Definition 2.19 *If G is a subterm permutation group with 1-context u and $u \equiv \Box$, then we say G is a* transitional *subterm permutation group. This is because the action of G on terms t in its domain cannot be obtained directly from its action on the proper subterms of t.*

Suppose that G is a subterm permutation group with 1-context $u[\ ]$ and the uniform set E_G of leaf permutative equations. Suppose that u is not $\Box$ and that the occurrence of $\Box$ in u is in the k-th subterm of u. Let u' be the k-th subterm of u. Then we define $G|_k$ to be the subterm permutation group with 1-context u' and the uniform set E_G of equations. Thus $(u, E_G)|_i = (u|_i, E_G)$. If u is $\Box$ or the occurrence of $\Box$ in u is not in the k-th subterm of u, then $G|_k$ is undefined. For general sequences α and β of integers, we define $G|_{\alpha\beta}$ as $(G|_\alpha)|_\beta$, and for the empty sequence λ, $G|_\lambda = G$.

Definition 2.20 *If $G = (E_1, \ldots, E_n)$ is a stratified subterm permutation group and labeled term $(t, u_1, \ldots, u_n)$ is in the domain of G, then we say $(t, u_1, \ldots, u_n)$ is a* transitional term *for G if at least one of the groups (u_i, E_i) is transitional, that is, some u_i is $\Box$. We note that at most one of the groups (u_i, E_i) can be transitional.*

Let $G = (E_1, \ldots, E_n)$ be a stratified permutation group. Suppose $(t, u_1, \ldots, u_n)$ is a labeled term in the domain of G. Suppose that t is not a transitional term for G. Let t_k be the k-th subterm of t, and let $(u_{i_1}, \ldots, u_{i_m})$ be the contexts whose occurrence of $\Box$ falls within t_k, that is, the k-th subterm of u_{i_j} contains the occurrence of $\Box$. Then $G|_k$ is defined to be the subterm permutation group $(E_{i_1}, \ldots, E_{i_m})$. Also, the labeled term $(t_k, u'_{i_1}, \ldots, u'_{i_m})$ is in the domain of $G|_k$, where u'_{i_j} is the k-th subterm of u_{i_j}. If t is a transitional term for G, then $G|_k$ is not defined. As before, for general sequences α and β of integers, we define $G|_{\alpha\beta}$ as $(G|_\alpha)|_\beta$, and for the empty sequence λ, $G|_\lambda = G$.

From now on, we will often deal with marked groups instead of labeled groups, since this avoids the necessity of specifically treating the contexts. Of course, it is easy to interchange between the two formalisms. For a marked group G, $G|_k$ is defined as above, but we need only say that a marked term $t|_k$ is in the domain of $G|_k$ if t is in the domain of G. Note that a marked term t is transitional if its topmost function symbol is marked.

Theorem 2.21 *Suppose G is a subterm permutation group for term t, with term t of the form $f(t_1, \ldots, t_n)$. Suppose G is not a transitional group. Then $S(t, G) = \{f(s_1, \ldots, s_m) : s_i \in S(t_i, G|_i), 1 \leq i \leq m\}$.*

Proof. Since t is not transitional for G, only one of the G_i will be different from the identity, and this G_i will permute subterms of t_i the same way as G did. $\square$

Theorem 2.22 *Suppose G is a (marked) stratified subterm permutation group and the marked term t is in the domain of G, where t is of the form $f(t_1, \ldots, t_m)$. Suppose t is not a transitional term for G. Then $S(t, G) = \{f(t'_1, \ldots, t'_m) : t'_i \in S(t_i, G|_i), 1 \leq i \leq m\}$.*

Proof. Each mapping ϕ in $\Phi(G)$ only influences one of the subterms $t|_i$ of t. The effect on $t|_i$ of such a ϕ is the same as the effect of a mapping in $G|_i$. Therefore any composition of mappings in G can be split up into the mappings that influence different subterms, as required by the theorem. Also, these mappings from different subterms commute, by theorem 2.13. $\square$

Corollary 2.23 *Suppose G is a (marked) stratified subterm permutation group and the marked term t is in the domain of G. Suppose t is not a transitional term for G. Let u be a context not containing any marked symbols, such that t is a context instance of u. That is, there are terms δ_i such that $t \equiv u[\delta_1, \ldots, \delta_k]$. Let α_i be the position of the i-th occurrence of $\square$ in u. Then $S(t, G) = \{u[t'_1, \ldots, t'_k] : t'_i \in S(\delta_i, G|_{\alpha_i}), 1 \leq i \leq k\}$*

Proof. By repeated application of the theorem, using induction on term depth. $\square$

We now apply previous results to say more about the structure of $S(t, G)$ for a stratified group G, in case t is a transitional term for G.

Corollary 2.24 *Suppose $G = ((E_1, g_1), \ldots, (E_n, g_n))$ is a (marked) stratified subterm permutation group and the marked term t is in the domain of G. Suppose t is a transitional term for the marked group $((E_1, g_1))$, that is, the top-level function symbol of t is the top-level symbol of E_1. Recall that $\Phi(E_1)$ is the set of generators of $((E_1, g_1))$. Let u be a context having occurrences of $\Box$ at all the (k) variable positions of r, where r is a redex for E_1. Let α_i be these positions of variables in r. Suppose $t \equiv u[\delta_1, \ldots, \delta_k]$. Then $S(t, G) = \{\phi[u(t'_1, \ldots, t'_k)] : \phi \in \Phi^*(E_1), t'_i \in S(\delta_i, G|_{\alpha_i}), 1 \leq i \leq k\}$.*

Proof. We know by theorem 2.13 that we can commute the ϕ in $\Phi(G)$ so that those from E_1 will be done last. Thus we can express any mapping from $\Phi^*(G)$ as $\alpha \cdot \beta$ where β is in $\Phi^*(((E_2, g_2), \ldots, (E_n, g_n)))$ and α is in $\Phi^*(E_1)$. Then $((E_2, g_2), \ldots, (E_n, g_n))$ is a subterm permutation group having t in its domain and t is not transitional for this group. Also, $\beta \in \Phi^*(((E_2, g_2) \ldots, (E_n, g_n)))$. Recall that t is $u[\delta_1, \ldots, \delta_k]$. The mapping β will act only within the terms δ_i, since u is essentially a redex of E_1, and the redexes of the various E_i in G do not overlap. This implies that u contains no marked symbols. Therefore, we can apply corollary 2.23 to the group $((E_2, g_2), \ldots, (E_n, g_n))$ and the context u to show the effect of β on t. Finally, the mappings α from $\Phi^*(E_1)$ (called ϕ in corollary 2.24) will apply by permuting the images t'_i of the terms δ_i, giving the desired result. $\Box$

3 Testing for Membership using Stratified Sets

We now consider how efficiently one can test set membership on sets of stratified terms, that is, given a term s and a pointed labeled group $((t, \overline{u}), G)$, to test whether $s \in S'((t, \overline{u}), G)$. This will be useful for lifting typical inferences to stratified sets, for example, rewriting all elements of a stratified set, or deleting equations that are already members of an existing pointed set. The membership algorithm will decompose this membership problem into simpler membership problems, whose solutions can then be obtained recursively and combined to obtain a solution to the original problem. By recording which subproblems have already been seen along with their solutions, this can be done in such a way as to avoid backtracking, that is, no subproblem needs to be considered more than once. We give a general algorithm, and also some special cases in which the problem may be solved faster.

For this problem, it is more convenient to work with marked terms rather than labeled terms. Let (t', G') be $m(((t, \overline{u}), G))$, that is, the corresponding pointed marked group. Suppose $G' = ((E'_1, g_1), \ldots, (E'_n, g_n))$. The problem becomes to test whether there exists a labeling $\overline{u}'$ such that the labeled term $(s, \overline{u}')$ is in $um^{G'}(S(t', G'))$, where (t', G') is a pointed marked group. The function $um^{G'}$ returns a labeled term; we are here interested in the term, but not the labels. So we define $um_1^{G'}$ so that if $um^{G'}(w) = (t, \overline{u})$, then $um_1^{G'}(w) \equiv t$. We then have the problem of testing whether s is in $um_1^{G'}(S(t', G'))$. We give a reasonable membership algorithm, using the following facts:

1. If the marked term $f(t_1, \ldots, t_k)$ is not transitional, then the term $g(s_1, \ldots, s_p) \in um_1^{G'}(S(f(t_1, \ldots, t_k), G'))$ iff $g = f$ and $p = k$ and for all i, $s_i \in um_1^{G'|_i}(S(t_i, G'|_i))$.

2. If $r[x_1, \ldots, x_p]$ is a redex for E'_j, the positions of the variables x_i in $r[x_1, \ldots, x_p]$ are α_i, and $s[x_1, \ldots, x_p]$ is $um_1^{G'}(r[x_1, \ldots, x_p])$, that is, a term like r except that the top-level symbol of s is unmarked, then the term $s[s_1, \ldots, s_p] \in um_1^{G'}(S(r[t_1, \ldots, t_p], G))$ iff there is a permutation γ of $\{1, 2, \ldots, p\}$ such that for all i, $s_i \in um_1^{G'|_{\alpha_i}}(S(t_{\gamma(i)}, G'|_{\alpha_i}))$ and if there exists $\phi \in \Phi^*(E'_j)$ such that $\phi(r[x_1, \ldots, x_p]) \equiv r[x_{\gamma(1)}, \ldots, x_{\gamma(p)}]$.

These two rules are all that is needed to compute the membership predicate. Both rules provide means for expressing the given membership problem in terms of simpler problems. The correctness of this algorithm follows from corollary 2.24. Some of the tests $s \in um_1^{G'}(S(t', G'))$ may be performed many times; if the results of these tests are cached to avoid repeated work, the work for the algorithm is at most proportional to the number of permutations in $\Phi^*(E'_j)$ times the size (number of distinct subterms) of s and is often faster, because the test $(\exists \phi \in \Phi^*(E'_j))(\phi(r[x_1, \ldots, x_p]) \equiv r[x_{\gamma(1)}, \ldots, x_{\gamma(p)}])$ can be done in polynomial time. This follows from the fact that the flat permutative consequence problem of theorem 1.1 has a polynomial time solution in that case. Thus if the number of γ is small, this problem has a polynomial time solution. In particular, we only need to consider γ such that $fr(s_i) = fr(t_{\gamma(i)})$ for all i. If s or t has no repeated subterms, then all the $fr(s_i)$ or $fr(t_i)$ will be disjoint (and hence distinct). This is related to the discussion in lemma 2.17. This means that there will be at most one γ such that $fr(s_i) = fr(t_{\gamma(i)})$ for all i. Only this one γ needs to be considered, which means that the membership problem can be solved in polynomial time if s or t has no repeated subterms. Also, if the sizes of the $\Phi^*(E_j)$ are bounded, that is, the number of variables in the equations are bounded, then by examining all elements of $\Phi^*(E_j)$ and all subterms of t and s, the algorithm can be executed in polynomial time. Otherwise, we can have the situation in which many terms $t|_\alpha$ for positions α in t can map onto the same subterm of s or the same subterm of s appears many places in s, and we then can obtain a problem at least as hard as graph isomorphism. (This will be shown later.) So there are still some difficult cases for this algorithm, but also some easy cases, and even the difficult cases can be done more efficiently than by explicitly generating all of the elements of S. One interesting case is when the contexts u_i of a labeled term $(t, \overline{u})$ are all independent, that is, no u_i is a context instance of any u_j, for $i \neq j$. In that case, all labeled terms $(s, \overline{u}') \in S((t, \overline{u}), G)$ will have the same independence property, in fact $\overline{u}' = \overline{u}$, and it is possible to perform the membership test and other operations faster.

3.1 Other operations

We can obtain an algorithm to test if s has an instance in $S'(t, G)$ in much the same way, by returning for each such s and t a set of such instances. Also, we can test if

s unifies with some term in $S'(t,G)$ by returning for each such s and t a set of most general unifiers. We can also test whether there is a term t' in $S'(t,G)$ having a subterm that unifies with s, et cetera. Then we can perform term rewriting, paramodulation, and critical pairs between a single equation $r = s$ and a set $S'(t,G)$ of terms. The details of these algorithms are relatively straightforward and not especially interesting; also, since we are really interested in operating on two sets $S(t_1,G^1)$ and $S(t_2,G^2)$ of terms, these algorithms involving a single equation and a set $S(t,G)$ of terms are of limited usefulness. Algorithms for operating on two stratified sets will be presented elsewhere.

4 Applications to Deduction

4.1 Lifting Operations to Stratified Sets

In order to apply stratified subterm groups to deduction, it will be necessary to operate directly on stratified sets rather than on individual clauses or equations. We now give some general comments about lifting operations from specific terms to stratified sets, and then consider a number of specific operations and how they may be performed efficiently on stratified sets. If t is a term and E a set of equations, let $S'_E(t)$ be $S'((t,u_1,\ldots,u_n),(E_1,\ldots,E_n))$ for a particular stratified set $E_1,\ldots,E_n$ and a particular labeling u_i such that $(t,u_1,\ldots,u_n)$ is in the domain of $(E_1,\ldots,E_n)$. We assume that $E \models E_i$ for all i and that this stratified set and labeling are obtained in some systematic manner from t and E.

We propose to lift operations in the following manner. Let us consider some operation F on pairs (t_1,t_2) of individual terms; for example, in order to find critical pairs, we may want to find subterms of t_2 that unify with t_1. Thus $F(t_1,t_2)$ represents a set of objects (such as terms, substitutions, positions) produced from the terms t_1 and t_2 via F. Then we want to lift this operation to stratified sets. We will not store all terms explicitly, but only stratified sets of terms. That is, we want to compute the union $F_U(t_1,t_2)$ of the sets $F(u_1,u_2)$ for all terms u_1 in $S'_E(t_1)$ and u_2 in $S'_E(t_2)$. Then we want to find stratified sets to represent $F_U(t_1,t_2)$, that is, we want to find a set $F_S(t_1,t_2)$ of terms such that $F_U(t_1,t_2)$ is the union over all t in $F_S(t_1,t_2)$, of $S'_E(t)$.

We can estimate the size of $F_S(t_1,t_2)$ in the worst case. Let M be the maximum cardinality of a set $F(u_1,u_2)$ for u_1 in $S'_E(t_1)$ and u_2 in $S'_E(t_2)$. Suppose the terms t_1 and t_2 both have length less than n, that is, they contain at most n occurrences of function and constant symbols. There are at most n^n (or possibly $n!$) terms in $S'_E(t_1)$ and $S'_E(t_2)$, because all such terms involve at most the n symbols in t_1 or t_2. (There can be $\Omega(n)$ distinct symbols, because there can be arbitrarily many distinct variables.) Therefore the cardinality of $F_U(t_1,t_2)$ is at most $M * n^n * n^n$ or $M * n^{2n}$ which, for small M, is $O(c^{n \log n})$. For common operations such as rewriting or forming critical pairs, M will be polynomial in n. Therefore the cardinality of $F_S(t_1,t_2)$ will be $O(c^{n \log n})$, and can be much smaller, since a large set of terms can be represented by a single stratified set. This is better than the situation for AC-unification, where the number of AC-unifiers can be double exponential [Dom92]. This is evidence that this use of stratified sets gives a better granularity of operation than AC-unification. It is known that AC-unification can be done in worst case double exponential time [KN92], which is at least an upper bound, if not a very small one.

We would like to compute $F_S(t_1, t_2)$ efficiently, that is, without computing the intermediates $F(u_1, u_2)$ explicitly. We pose this as an interesting problem, without making an attempt to present an efficient algorithm for it. However, we have developed such algorithms, and plan to present them elsewhere.

We note that this approach differs from specialized E-unification algorithms in that we do not apply any additional substitutions to the terms; the operations performed correspond exactly to those that would have been performed on the original set of clauses or equations. This means that our granularity of operations is finer than for E-unification, in a sense. However, it is also coarser, since our approach can incorporate arbitrary leaf permutations into the algorithms. Since AC-unification can produce a double exponential number of unifiers [Dom92], it may be that AC-unification has too coarse a granularity, that is, the inference steps are too large.

4.2 Stratified clause sets

So far, we have mainly considered sets of terms. However, when performing deduction, we also need to consider equations and clauses. We now sketch how stratified subterm permutations can be applied to clauses. Suppose that $C[t_1, \ldots, t_m]$ is an equation or clause and the t_i are its maximal subterms. Suppose $\mathcal{G}$ is a set of pointed groups of the form $\{((t_1, \overline{u}^1), G_1), \ldots, ((t_m, \overline{u}^m), G_m\ )\}$. Then we can define $S'(C[t_1, \ldots, t_m], \mathcal{G})$ to be $\{C[v_1, \ldots, v_m] : v_i \in S'((t_i, \overline{u}^i), G_i)\}$. We call this a *stratified clause set.* We note that for an arbitrary clause $D \in S'(C[t_1, \ldots, t_m], \mathcal{G})$, $E \models (C[t_1, \ldots, t_m] \equiv D)$, so if $C[t_1, \ldots, t_m]$ has been derived, then all such clauses D are in principle derivable, also. We represent this stratified clause set by the pair $(C[t_1, \ldots, t_m], \mathcal{G})$, which we call a *pointed stratified clause set*, or just a *pointed set.* We propose to process the stratified clause sets $S'(C[t_1, \ldots, t_m], \mathcal{G})$ of equations or clauses directly rather than processing their individual elements separately. At any time, we will have a set S of clauses and a set *Pairs* of pointed clause sets. The pointed sets $(C, \mathcal{G})$ are considered as compact representations for all of the elements $S'(C, \mathcal{G})$. Initially, each clause C has the trivial pointed set $\mathcal{G}$, so that $S'(C, \mathcal{G}) = \{C\}$. Then, we perform inferences to generate new clauses, and also look for new stratifications of existing clauses relative to the set E of equations that have been derived so far. Each inference needs to be performed between two sets $S'(C_1, \mathcal{G}^1)$ and $S'(C_2, \mathcal{G}^2)$ rather than between two clauses. This therefore involves lifting operations (such as resolution or paramodulation) from individual clauses to stratified clause sets. We have not presented the details of how this can be done, but the techniques presented in section 4.1 are relevant here.

We also need a deletion criterion, to know when a pointed clause set is redundant, and can be removed. For this, suppose that we have the two pointed groups $(C_1, \mathcal{G}^1)$ and $(C_2, \mathcal{G}^2)$. Let S_1 be $S'(C_1, \mathcal{G}^1)$ and let S_2 be $S(C_2, \mathcal{G}^2)$. We present a simple, partial test for the subset relation $S_1 \subseteq S_2$. Let us make the top-level subterms of C_1 and C_2 explicit, as, $C_1[v_1, \ldots, v_m]$ and $C_2[w_1, \ldots, w_p]$. We will not deal with clause subsumption here, so we only need to consider the case in which $m = p$. First, we test whether $C_1 \in S(C_2, \mathcal{G}^2)$. For this, we look for a permutation $y_1, \ldots, y_m$ of the variables $x_1, \ldots, x_m$ so that $C_1[x_1, \ldots, x_m]$ and $C_2[y_1, \ldots, y_m]$ are identical. This can be nontrivial to find, because the literals in clauses can be permuted; in fact, there may be more than one such permutation. For example, suppose that C_1 is $P(s_1) \vee P(s_2)$ and C_2 is $P(t_1) \vee P(t_2)$. Then the maximal subterms of C_1 are s_1 and s_2, and the maximal subterms of C_2 are t_1 and t_2. There are two possible permutations that permit the s_i to correspond to the t_j.

Let us fix our attention on one such permutation, and for simplicity and without loss of generality suppose that it is the identity. This means that $C_1[x_1, \ldots, x_m]$ and $C_2[x_1, \ldots, x_m]$ are identical. Let $\mathcal{G}^1$ be $\{((v_1, \overline{u}^1), G_1^1), \ldots, ((v_m, \overline{u}^m), G_m^1)\}$ and let $\mathcal{G}^2$ be $\{((w_1, \overline{q}^1), G_1^2), \ldots, ((w_m, \overline{q}^m), G_m^2)\}$. Then $S_1 \subseteq S_2$ if for all i, $S'((v_i, \overline{u}^i), G_i^1) \subseteq S'((w_i, \overline{q}^i), G_i^2)$. This is an instance of the general problem of testing if one stratified set of terms is a subset of another stratified set of terms. Therefore, we now consider this problem separately; this also permits us to simplify the notation.

Suppose therefore that v and w are two terms and $\overline{u}$ and $\overline{q}$ are two labelings of them, and that G^1 and G^2 are two subterm permutation groups such that $(v, \overline{u})$ is in the domain of G^1 and $(w, \overline{q})$ is in the domain of G^2. We do not present a complete test for $S'((v, \overline{u}), G^1) \subseteq S'((w, \overline{q}), G^2)$, but we give a sufficient condition for this. Let T_1 be $S'((v, \overline{u}), G^1)$ and let T_2 be $S'((w, \overline{q}), G^2)$. To test whether $T_1 \subseteq T_2$, we first test whether $v \in T_2$. This can be done by the membership algorithm just presented. If this fails, we know that T_1 is not a subset of T_2. If this succeeds, then there is a $\overline{u}^2$ such that $(v, \overline{u}^2) \in T_2$, which can be obtained from the computation of the membership algorithm. We note then that $S'((v, \overline{u}^2), G^2) = T_2$ and $S'((v, \overline{u}), G^1) = T_1$. Therefore we have two pointed subterm groups with a common head term.

We now compare the pointed groups $((v, \overline{u}), G^1)$ and $((v, \overline{u}^2), G^2)$. Let G^1 be $(E_1^1, E_2^1, \ldots, E_{n_1}^1)$ and let G^2 be $(E_1^2, E_2^2, \ldots, E_{n_2}^2)$. We now test whether for all i there exists j such that $u_j \equiv u_i^2$ and $\Phi^*(G_i^1) \subseteq \Phi^*(G_j^2)$, where G_i^1 is the subterm permutation group (u_i^2, E_i^1) and G_j^2 is the subterm permutation group (u_j, E_j^2); in this case, it follows easily that $T_1 \subseteq T_2$. The test $\Phi^*(G_i^1) \subseteq \Phi^*(G_j^2)$ can be done in polynomial time, by testing whether the redexes of E_i^1 and E_j^2 are variants and $\Phi(G_i^1) \subseteq \Phi^*(G_j^2)$; this in turn can be done by testing each generator of G_i^1 for membership in $\Phi^*(G_j^2)$. To test whether $\phi_{r,s} \in \Phi^*(G_j^2)$, it suffices to test whether $s \in S'((r, \Box), E_j^2)$. This test for each generator can be done in polynomial time, by theorem 1.3.

4.3 Choosing a Stratification

In order to apply stratified subterm permutation groups to deduction, we need a method for choosing one of the many possible groups. Suppose that we have derived a set $\{C_1, \ldots, C_m\}$ of clauses. The object is to find stratified clause sets including as many of these clauses as possible, to achieve the greatest possible saving of work. Suppose C_i

is $C_i[v_1^i, \ldots, v_p^i]$, where the v_j^i are the maximal subterms of C_i, and for simplicity we only consider clauses having the same number of maximal subterms (since each number can be treated separately). We want to find a set $\mathcal{G}$ of pointed stratified subterm groups and a clause $C \in \{C_1, \ldots, C_m\}$ such that $S'(C, \mathcal{G})$ includes as many as possible of the clauses C_i. For this purpose, we want to find pointed stratified subterm groups $((v_j, \overline{u}^j), G_j)$ where v_j is one of the terms v_j^i which include as many elements as possible from the set $\{v_j^1, v_j^2, \ldots, v_j^m\}$ of terms. Then we can let $\mathcal{G}$ be the set $((v_1, \overline{u}^1), G_1), \ldots, ((v_p, \overline{u}^p), G_p)$, which will include many of the clauses $C_1, \ldots, C_m$. For example, if the C_i are equations, then we may have a set $\{r_1 = s_1, \ldots, r_m = s_m\}$ of equations. Then we want to find groups $((v_1, \overline{u}^1), G_1)$ and $((v_2, \overline{u}^2), G_2)$ so that $v_1 \in \{r_1, r_2, \ldots, r_m\}$ and $v_2 \in \{s_1, s_2, \ldots, s_m\}$ and $S'((v_1, \overline{u}^1), G_1)$ contains as many elements as possible from the set $\{r_1, r_2, \ldots, r_m\}$ and $S'((v_2, \overline{u}^2), G_2)$ contains as many elements as possible from the set $\{s_1, s_2, \ldots, s_m\}$. Then $S'(v_1 = v_2, \{((v_1, \overline{u}^1), G_1), ((v_2, \overline{u}^2), G_2)\})$ should include many of the equations $r_i = s_i$. In general, the relevant problem becomes the following: Given a set $\mathcal{T}$ of terms, to find a stratified subterm permutation group including as many of the elements of $\mathcal{T}$ as possible.

The goal is to approximate the E-equivalence classes as closely as possible by stratified sets, where E is the set of equations. For deductive purposes, we only need to consider terms that are in normal form with respect to ordered rewriting; thus, we need only be concerned with E-equivalence classes of terms in normal form. This means, for example, that we need not consider terms having subterms of the form $(r * s) * t$, assuming that this subterm can be rewritten to $r * (s * t)$ under ordered rewriting. Also, we prefer to avoid creating two stratified sets with a non-empty intersection, in order to minimize the number of stratified sets considered. We know that the stratified sets are always subsets of the E-equivalence classes. Therefore, in order to approximate E-equivalence, we want to make the stratified sets as large as possible. Of course, it is not always possible to completely capture the E-equivalence classes by stratified sets. In addition, computational limitations restrict what can feasibly be attained in this direction.

We first give an idealized approach to this problem, and then make it more practical. Suppose that t is a term; we want to find subterm permutation groups with t in their domains. Then we can combine a number of such groups to obtain a stratified subterm permutation group with t in its domain. The general idea for finding individual subterm permutation groups for t is to choose a linear term r and a 1-context u such that for some Θ, $t \equiv u[r\Theta]$, that is, some subterm of t is an instance of r. Then we find the set $\{s_1, s_2, \ldots, s_k\}$ of variants of r such that $r = s_i$ is a leaf permutation and such that $E \models r = s_i$ for $1 \leq i \leq k$. Let E^r be this set of equations $\{r = s_1, r = s_2, \ldots, r = s_k\}$; then (u, E^r) is a subterm permutation group with t in its domain.

We note that determining E^r from r can be very difficult, because it requires equational theorem proving. Therefore, we present a more practical approach. We define $E^{r,0}$ to be the set of equations $\{r = s : s \textit{ is a variant of } r, r \rightarrow_E s\}$. Then $E^{r,0}$ is essentially a set of equations from E, possibly instantiated, and possibly with context added. Also, $E^{r,0}$ may easily be computed from E and r. Furthermore, with u a 1-context and r a linear term such that for some Θ, $t \equiv u[r\Theta]$, $(u, E^{r,0})$ is a subterm permutation group for t. Let E_1 be E together with some equations that are logical consequences of E, possibly generated by superpositions or other kinds of inferences. Then $(u, E_1^{r,0})$ is also a subterm permutation group for t, and may include more equations. Using E_1 instead of E is therefore desirable.

We still have the problem of finding likely u and r. The goal is to find expressions of the form $\{C[v_1, \cdots, v_n] : v_1 \in S'(t_1, G_1), \cdots, v_n \in S'(t_n, G_n)\}$ or of the form $\{v_1 = v_2 : v_1 \in S'(t_1, G_1), v_2 \in S'(t_2, G_2)\}$ that include as many equations or clauses as possible that have been generated so far in a proof attempt. Therefore it makes sense to examine the equations produced during an attempted proof, and look for regularities among them. Typically in the running of a completion procedure, uniform sets of equations will be generated. Suppose $r = s$ is one member of a uniform set of equations. Let u be a 1-context such that for some Θ, $t \equiv u[r\Theta]$. Then the subterm permutation group $(u, E^{r,0})$ is nontrivial and is a likely candidate to be considered. For example, we may generate the following set E_2 of uniform equations as logical consequences of E:

$$f(x, f(y, f(z, w))) = f(y, f(x, f(w, z)))$$

$$f(x, f(y, f(z, w))) = f(z, f(y, f(x, w)))$$

$$f(x, f(y, f(z, w))) = f(w, f(z, f(y, x)))$$

Consider the term $t \equiv$

$$g(f(a, f(b, f(g(a), g(b)))))$$

which may be the maximal term in an equation or clause. This term has the subterm $f(a, f(b, f(g(a), g(b))))$ which is an instance of the left (or right)-hand sides of the above uniform set of equations. Thus we can choose r as the term $f(x, f(y, f(z, w)))$ and the context u as $g(\Box)$. Then for some Θ, $t \equiv u[f(x, f(y, f(z, w)))\Theta]$, that is, $t \equiv g(f(x, f(y, f(z, w))))\Theta$. Therefore $(u, E_2^{r,0})$ is a subterm permutation group for t, and E_2 is a subset of $E_2^{r,0}$. In this way we obtain the subterm permutation group G generated by the elements that interchange the subterms $a, b, g(a), g(b)$ of t according to the uniform equations given above. Such uniform sets of equations frequently arise, especially with associative-commutative operators. Also, we note that any set of flat permutations is uniform.

The subterm permutation groups constructed in this way may actually include more equations than those they were constructed from. That is, E_2 may be a proper subset of $E_2^{r,0}$. This means that these groups will be even better approximations to the E-equivalence relation. This can happen because there may be non-uniform equations that act on u as if they were uniform. For example, the equation

$$f(x, f(y, z)) = f(y, f(x, z))$$

can be used to rewrite the term t, producing the term

$$g(f(b, f(a, f(g(a), g(b)))))$$

In this way we obtain a ϕ_1 mapping terms of the form $g(f(u_1, f(u_2, f(u_3, u_4))))$ onto terms of the form $g(f(u_2, f(u_1, f(u_3, u_4))))$. We can also apply this equation elsewhere to obtain the mapping ϕ_2 from terms of the form $g(f(u_1, f(u_2, f(u_3, u_4))))$ to terms of the form $g(f(u_1, f(u_3, f(u_2, u_4))))$. The equation $f(x, f(y, z)) = f(y, f(x, z))$ together with E_2 is not uniform; that is, $E_2 \cup \{f(x, f(y, z)) = f(y, f(x, z))\}$ is not uniform. However, these mappings ϕ_1 and ϕ_2 correspond to the equations

$$f(x, f(y, f(z, w))) = f(y, f(x, f(z, w)))$$

$$f(x, f(y, f(z, w))) = f(x, f(z, f(y, w)))$$

which, together with E_2, are uniform. These equations are included in $E_2^{r,0}$. Also, if f is an associative operator, then we can associate f to the right in a term t, obtaining subterms of t of the form $f(t_1, f(t_2, \ldots, f(t_{n-1}, t_n)\ldots))$. Then the subterm r given by $f(x_1, f(x_2, \ldots, f(x_{n-1}, x_n)\ldots))$ is a good candidate redex for a subterm permutation group, especially if f is also commutative.

We have just seen how a subterm permutation group for t may be obtained from E. To obtain a stratified subterm permutation group, we look for a set of such subterm permutation groups $(u_j, E^{r_j,0})$ for various u_j and r_j such that their occurrences r_j of redexes do not overlap. Then we obtain the labeled term $(t, u_1, \ldots, u_n)$ in the domain of the stratified subterm permutation group $(E^{r_1,0}, E^{r_2,0}, \ldots, E^{r_n,0})$.

We now give a simple algorithm for this. Suppose t is a term and $G_1, G_2, \ldots, G_n$ are subterm permutation groups on t. We want to find a subset of the G_i that can be made into a stratified set of permutation groups and a labeling for t. One algorithm is to exhaustively consider all subsets and select one of them (or all of them) that are stratified. It is possible to do a little better than this, by noting that if a set of groups is not stratified, then no superset will be either. So one can begin with one element subsets of $G_1, G_2, \ldots, G_n$, then construct two element subsets, etc., discarding those that are not stratified and continuing to add elements one by one to those that remain.

More formally, we first consider all pairs (G_i, G_j) of groups and for each one we determine whether G_i and G_j are *compatible.* We say that two (non-stratified) subterm permutation groups (u_1, E_1) and (u_2, E_2) are compatible if the occurrences of $\Box$ in u_1 and u_2 are in independent positions, or if the redexes do not overlap, that is, u_2 may be expressed as $u_1[r_1\Theta]$ or u_1 may be expressed as $u_2[r_2\Theta]$, where Θ is a substitution and r_i are redexes for E_i. Note that the terms u_i each contain one occurrence of $\Box$. Then the algorithm is as follows, where **choose** indicates a nondeterministic choice:

```
$H \leftarrow \{G_1, G_2, \ldots, G_n\}$;
$G \leftarrow \{\}$;
while ($H$ is not empty) do
        choose $G'$ in $H$;
        remove $G'$ from $G$;
        if $G'$ is compatible with all elements of $G$ then
                add $G'$ to $G$ fi
od;
```

The result of this algorithm is a set of compatible (non-stratified) subterm permutation groups for t. These can then be combined into a labeling for t and a stratified subterm permutation group G. This algorithm can be used to find a "good" stratification by choosing G' in some heuristic manner, say, choosing G' having the largest number of generators. It can also be used exhaustively to generate all stratifications of a term t. It is possible to refine this algorithm so that the disjointness condition is satisfied, that is, if G_1 is the stratified group generated for term t_1 and $t_2 \in S(t_1, G)$, then the group G_2 generated for t_2 will satisfy $S(t_2, G_2) = S(t_1, G_1)$. This is possible because t_2 will contain essentially the same redexes as t_1, but in different positions. Such a group G_2 can then be deleted, since it contributes nothing new. We plan to give the details elsewhere.

We note that if G and G' are two stratified subterm permutation groups, and t is a labeled term in their domains, and $S(t, G) \subseteq S(t, G')$, then typically we will choose to consider the pair (t, G') and ignore (t, G). Thus, if one group has strictly more redexes than another or sets E_i with strictly more equations, then it is preferable to the other group. These observations have the consequence that if E is a set of flat permutations, then we only need to consider essentially one stratification; for a term t, for each 1-context u and each m-context $w \equiv f(\Box, \ldots, \Box)$, the group $(u, E^{r,0})$ will be included in the stratified set if for some Θ, $t \equiv u[f(x_1, \ldots, x_n)\Theta]$. That is, all subterm permutation groups may be used together in this case, since they will all be compatible because E is a flat permutative theory. A similar case occurs when t contains associative-commutative operators; such operators can be "flattened," which essentially makes the associative-commutative equations flat permutations.

Now, there may be various ways to stratify a term, and one of them must be chosen in some way. There is nothing to prevent us from using more than one stratification of the same term, too. However, in common cases, the stratification will be obvious; for example, if we have a polynomial written as a sum of products, where the summation and product operator are associative and commutative, then the lowest groups would permute variables within the products (e.g., $x * y + z * w$ is mapped to $y * x + z * w$) and the highest group would permute the products with one another (e.g., $x * y + z * w$ is mapped to $z * w + x * y$). We note that none of the kinds of permutations defined in [Fro89] directly include stratified subterm permutation groups except for the *symbol-preserving* class which is more general even than stratifications and is not extensively treated in [Fro89]. There may also be permutations that apply to t but that do not fit into the chosen stratification. These might have to be dealt with explicitly, that is, used as arbitrary equations or rewrite rules would be. However, permutations that are incorporated into the stratification are implicit and need not be handled explicitly. After each application of an explicit rewrite rule, a stratification of the new resulting term can be done after reducing the term to normal form.

5 Specific Results

We now present a variety of results about the complexity of various problems involving permutative equations. Of course, there are also many problems that we have not considered. We say a term (or clause) is *ground* if it contains no variables.

5.1 Flat Permutative Equations

We first state a general result that will be useful a number of times later.

Definition 5.1 *A term s is* flat *if it is of the form $f(s_1, \ldots, s_n)$ for some f and the s_i are variables or constant symbols. If $n = 0$, then we obtain as a special case that a constant symbol is a flat term. An equation $s = t$ is flat if s and t are flat terms.*

Theorem 5.2 *If $E \models s = t$ for a set E of flat permutative equations and a flat equation $s = t$, then there is a flat permutation $u = v$ such that $E \models u = v$ and $s = t$ is an instance of $u = v$.*

Proof. Given a set E of equations, let R^E be a term-rewriting system containing the rule $r_1 \rightarrow r_2$ for each equation $r_1 = r_2$ in E. By Birkhoff's theorem, $E \models s = t$ iff $s \leftrightarrow^*_{R^E} t$. If s and t are flat, then this implies that all the applications of rewrite rules in this derivation between s and t are at the top level. By combining them together, we obtain a flat permutation $u = v$ as in the theorem. $\square$

Corollary 5.3 *If $E \models s = t$ for a flat permutative theory E and a flat equation $s = t$, then this can be verified in nondeterministic polynomial time.*

Proof. Such a logical consequence $u = v$ must exist, by the theorem. However, the problem whether $E \models u = v$ for such a flat permutative equation $u = v$ and a flat permutative theory E is solvable in polynomial time, by theorem 1.1. This is done by nondeterministically generating a flat permutative logical consequence $u = v$ and verifying that $s = t$ is an instance of $u = v$ and that $E \models u = v$. $\square$

For non-flat equations we have a similar result.

Theorem 5.4 *Suppose s and t are terms of the form $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$, respectively. Suppose E is a flat permutative theory. If $E \models s = t$, then there is a permutation ϕ such that $E \models s_i = t_{\phi(i)}$ for all i and such that $E \models f(x_1, \ldots, x_n) = f(y_1, \ldots, y_n)$ where $y_{\phi(i)} \equiv x_i$.*

Proof. By Birkhoff's theorem, again. We consider the structure of the derivation $s \leftrightarrow^* t$. We note that the order of applications of rules at the top level and to proper subterms can be interchanged, so that all the rewrite rules are applied to subterms first. This essentially divides the derivation into an initial part, involving proper subterms, and a final part, involving the top level. The final part determines ϕ as shown in theorem 5.2, and the initial part involves demonstrations that $E \models s_i = t_{\phi(i)}$ for all i. $\square$

Corollary 5.5 *If $E \models s = t$ for a flat permutative theory E, then this can be verified in nondeterministic polynomial time.*

Proof. By induction, one can construct nondeterministic polynomial length demonstrations that $E \models s_i = t_{\phi(i)}$ for all i. As in corollary 5.3, one can also verify that $E \models f(y_{\phi(1)}, \ldots, y_{\phi(m)}) = f(y_1, \ldots, y_m)$ in nondeterministic polynomial time by exhibiting the flat permutative logical consequence $u = v$ and verifying that $f(y_{\phi(1)}, \ldots, y_{\phi(m)}) = f(y_1, \ldots, y_m)$ is an instance of $u = v$ and that $E \models u = v$. $\square$

Corollary 5.6 *Suppose E is a flat permutative theory and t is a term. Then there is a stratification $\{G_1, \ldots, G_n\}$ of t such that for all terms u, $u \in S(t, G_1, \ldots, G_n)$ iff $E \models t = u$.*

Proof. Suppose $t|_\alpha$ is of the form $f(t_1, \ldots, t_m)$ for some f and m. Then let $\mathcal{P}_\alpha$ be the set of positions $\alpha 1, \ldots, \alpha m$. Consider the subterm permutation group $G_{\mathcal{P}_\alpha, E}(t)$ which permutes the subterms t_i. Let $\{G_1, \ldots, G_n\}$ include all such groups $G_{\mathcal{P}_\alpha, E}(t)$ for all positions α of function symbols in t. By theorem 5.4, such permutations are enough to prove any logical consequence of E. Since we can apply them innermost first, the corollary follows. $\square$

The implication of this corollary is that it is only necessary to consider this one stratification of t, since it captures the E equivalence class of t by itself.

Theorem 5.7 *Consider the problem of deciding whether $E \models s = t$, where E is a flat permutative theory and $s = t$ is a flat ground equation. This problem is as hard as graph isomorphism. However, this problem is still in NP. For this it suffices to have at most two equations in E.*

Proof. We show this result by reduction from graph isomorphism, as follows: Suppose H_1 and H_2 are two directed graphs over n vertices. Consider a term of the form $f(c_1, c_2, \ldots, c_{n^2})$ where each c_i is either 0 or 1, according to whether a particular edge is in H_1 or not. Thus we have some bijection $h(x, y)$ of the n^2 possible edges (i, j) onto the integers $h(i, j)$ between 1 and n^2. For example, we may take $h(i, j)$ to be $i + n * (j - 1)$. We then consider two permutations of the set $\{1, 2, \ldots, n^2\}$; p_1 exchanges 1 and 2, and p_2 rotates the integers, that is, $p_2(i) = i + 1$ if $i < n^2$, and $p_2(n^2) = 1$. Then all other permutations of this set of integers can be obtained as compositions of these two. Now, we have induced permutations p_1' and p_2' on the set $\{1, 2, \ldots, n^2\}$ defined so that $p_1'(h(a, b)) = h(p_1(a), p_1(b))$ and $p_2'(h(a, b)) = h(p_2(a), p_2(b))$. We then have the two flat permutations $f(x_1, \ldots, x_{n^2}) = f(x_{p_1'(1)}, \ldots, x_{p_1'(n^2)})$ and $f(x_1, \ldots, x_{n^2}) = f(x_{p_2'(1)}, \ldots, x_{p_2'(n^2)})$. We consider the equation $s_1 = s_2$ where s_i are obtained by encoding the graphs H_i as indicated above, that is, the $h(i, j)$ argument of s_k is 1 if the edge (i, j) is present in H_k, and 0 otherwise. Then, this equation $s_1 = s_2$ is a logical consequence of the two flat permutations given above, iff H_1 and H_2 are isomorphic. This is so because of the correspondence between permutations of the arguments of f and permutations of the nodes of the graphs, and because of the fact that all permutations of the set $\{1, 2, \ldots, n^2\}$ are generated by p_1 and p_2. Therefore, testing if a given equation is a logical consequence of two flat permutations is as hard as graph isomorphism, which is not known to be polynomial. Membership in NP follows from corollary 5.3. □

This result is unexpected, because of the result from theorem 1.1 that there is a polynomial time algorithm to decide if $E \models s = t$, where E is a flat permutative theory and $s = t$ is a flat permutation. Also, the same problem in which all the arguments of f are distinct, is polynomial, as shown in corollary 1.2. Theorem 5.7 depends on the existence of function symbols of large arity. If the arity is bounded but we have leaf permutations, then we can still get the preceding result by encoding larger terms using trees.

Theorem 5.8 *The equational theory of a set of two (or more) leaf permutative equations is as hard as graph isomorphism. This is still true when the arity of all function symbols are bound to two.*

Proof. We use the same result as for the preceding theorem, encoding the graph permutations using leaf permutations. For example, we can encode a term $f(c_1, c_2, \ldots, c_{n^2})$ using the term $k(c_1, k(c_2, \ldots, k(c_{n^2-1}, c_{n^2}) \ldots))$ where k is a binary function symbol. Then the equational theory for a theory of only two leaf permutations can still be as hard as graph isomorphism, by the same reduction as above, adapted to this encoding using k. However, we do not know whether this problem is in NP; it is clearly in polynomial space becaues the sizes of the terms does not grow. □

5.2 Limiting Repetitions

We note that the result above concerning two permutations, depends on the fact that the same constant symbol may appear in many positions. It is natural to ask if one can get a better result if one restricts the number of repetitions of constant symbols among the arguments of terms. Indeed, this is possible.

Theorem 5.9 *Suppose E is a flat permutative theory and $s = t$ is a flat equation. Suppose that s is $f(s_1, \ldots, s_n)$ and t is $f(t_1, \ldots, t_n)$. Suppose that there are at most k repetitions of arguments in s and t. That is, the set of i such that for some $j \neq i$, $s_i = s_j$, has at most k elements, and similarly for the subterms t_i. Then we can decide whether $E \models s = t$ in time exponential in k. Also, this problem is in NP. For fixed k, the time bound is polynomial.*

Proof. We note as before that it suffices to consider the case in which s and t are ground terms. Let Φ be the set of permutations ϕ such that ϕ maps the arguments of s to the arguments of t. Thus $t_{\phi(i)} \equiv s_i$ for all i. We note that the number of elements in Φ is at most k^k, since there are at most k repetitions of arguments in s and t. Therefore each of the (at most k) arguments of s that are repeated, can map via ϕ onto one of the (at most k) arguments of t that are repeated. This permits at most k^k choices. Let G be the permutation group corresponding to E. Each equation in E corresponds to a permutation, and G is the group generated by these permutations. For each ϕ in Φ, we can test if ϕ is in G, and this can be done in polynomial time by theorem 1.3. Also, we essentially showed in theorem 5.2 that $E \models s = t$ iff there exists $\phi \in \Phi$ such that $\phi \in G$. Membership of this problem in NP follows from the membership of the problem of theorem 5.7 in NP. □

The implication of this result is that even though the equational theory of E is as hard as graph isomorphism, there are still some easily decidable parts of this theory, namely, equations with few repetitions. This gives us a smooth transition between the polynomial time algorithm for the flat permutative consequence problem of theorem 1.1 and theorem 5.7 above.

The same result extends to non-flat terms s and t and a flat permutative theory E.

Theorem 5.10 *Suppose E is a flat permutative theory and s and t are arbitrary (not necessarily flat) terms. Suppose the number of repetitions of subterms in s and t is at most k. Then one can decide if $E \models s = t$ in time exponential in k.*

Proof. The argument is essentially the same as above, namely, one constructs a set Φ of permutations having a number of elements exponential in k, and one shows that $E \models s = t$ iff some element of Φ is in G. However, it is necessary to process the subterms first recursively. Suppose E is a flat permutative theory and s and t are terms with only a finite number of repetitions among their subterms. Suppose s is $f(s_1, \ldots, s_m)$ and t is $f(t_1, \ldots, t_m)$. Suppose $E \models s_i = t_j$; then it must be that s_i and t_j have a common subterm (in fact, a variable or a constant symbol), since E consists only of permutations. Therefore, the number of j such that $E \models s_i = t_j$ is at most k. We can therefore solve this problem in time exponential in k by recursively computing for the subterms s_i and t_j whether $E \models s_i = t_j$ and then looking for a permutation ϕ such that for all i, $E \models s_i = t_{\phi(i)}$ and $E \models f(y_{\phi(1)}, \ldots, y_{\phi(m)}) = f(y_1, \ldots, y_m)$. This suffices by theorem 5.4. The number of such ϕ is exponential in k, as mentioned before, and the test whether $E \models f(y_{\phi(1)}, \ldots, y_{\phi(m)}) = f(y_1, \ldots, y_m)$ can also be done in polynomial time, since this is the flat permutative consequence problem of section 1. □

5.3 Bounded Arity Function Symbols

We now consider what happens if we only allow bounded arity function symbols and only flat permutative equations. Recall from theorem 5.7 that deciding if a flat equation follows from a flat permutative theory is as hard as graph isomorphism, if there is no bound on the arities. For bounded arities, the problem is easier, as the following theorem shows.

Theorem 5.11 *Suppose all function symbols have bounded arity and E is a flat permutative theory. Then the equational theory of E is decidable in polynomial time.*

Proof. The idea is to decide E-equivalence from the bottom up, using theorem 5.4. That is, to determine if an equation $s = t$ is a logical consequence of E, we can replace the variables in s and t by new constant symbols, obtaining ground terms s' and t'. Then $E \models s = t$ iff $E \models s' = t'$. We therefore give a method for solving this latter problem. Suppose s' is $f(s_1, \ldots, s_m)$ and t' is $g(t_1, \ldots, t_n)$. If $f \not\equiv g$ or $m \neq n$ then $E \not\models s' = t'$. Otherwise, we recursively test for all i and j whether $E \models s_i = t_j$. Also, we test for all i and j whether $E \models s_i = s_j$ and whether $E \models t_i = t_j$. This latter information suffices to divide the subterms s_i and t_j into E-equivalence classes. We replace each equivalence class by a distinct new constant symbol c_k. Notice that it is possible that s_i and t_j are in the same equivalence class. We thus obtain two terms $s'' \equiv f(c_{a_1}, \ldots, c_{a_n})$ and $t'' \equiv f(c_{b_1}, \ldots, c_{b_n})$ where $c_{a_i} \equiv c_{a_j}$ iff $E \models s_i = s_j$, $c_{b_i} \equiv c_{b_j}$ iff $E \models t_i = t_j$, and $c_{a_i} \equiv c_{b_j}$ iff $E \models s_i = t_j$. Then we have that $E \models s'' = t''$ iff $E \models s' = t'$; this follows because the theory is flat, and all terms s_i and t_j can be replaced by unique representatives of their equivalence classes. Finally, the problem of whether $E \models s'' = t''$ is solvable in polynomial time, by theorem 5.2 and by the fact that there are only a bounded number of permutations to consider, since the arity is bounded. Since s'' and t'' are flat, and of bounded arity, we can exhaustively enumerate all terms u such that $E \models s'' = u$ in polynomial time, and test whether t'' is included in this list of terms. The cost is still exponential in the arity, however, and this seems inevitable because of theorem 5.7 concerning graph isomorphism. The correctness of this algorithm follows from theorem 5.4. The polynomial time bound follows from the fact that the recursive calls will all be of the form $E \models s_1 = t_1$ where s_1 is a subterm of s or t and t_1 is a subterm of s or t; the number of such calls is quadratic, and each such recursive call will be executed at most once. $\square$

5.4 A Single Flat Permutation

We have still not settled the case of an equational theory E with flat permutative rules of unbounded arity but only one permutation. For this case, it turns out that the equational theory can be decided in polynomial time, but the proof is somewhat complicated.

Theorem 5.12 *Suppose E is an equational theory consisting of a single flat permutative equation. Then the equational theory of E can be decided in polynomial time.*

Proof. We first consider the problem whether $E \models s = t$ where s and t are flat. Suppose that E contains just the one equation $f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)$ where the x_i are all distinct and the y_i are a permutation ϕ of the x_i. Then this permutation can be expressed as a product of disjoint cycles. Since the order of the arguments to f is immaterial (as long as the order of the x_i and y_i is changed in the same way), we might as well assume that the cycles are all of the form $C_i = (x_{a_i}, x_{a_i+1}, \ldots, x_{b_i})$ for various a_i and b_i. That is, $y_{a_i+1} \equiv x_{a_i}$, $y_{a_i+2} \equiv x_{a_i+1}$, and so on, and $y_{a_i} \equiv x_{b_i}$. We are also given an equation $s = t$ and wish to determine if $E \models s = t$. For this it suffices to consider the case in which s and t are ground terms, and we first consider the case in which they are flat ground terms, that is, $s = t$ is of the form $f(c_1, c_2, \ldots, c_n) = f(d_1, d_2, \ldots, d_n)$ where the c_i and d_j are (not necessarily distinct) constant symbols. The application of the equation $f(x_1, x_2, \ldots, x_n) = f(y_1, y_2, \ldots, y_n)$ to s permutes the c_i. Since there is only one equation, $E \models s = t$ iff the sequence d_i can be obtained from the sequence c_i by some number of applications of the permutation ϕ. For this, we can consider separately the action of each cycle C_j on the c_i, and determine for which integers m we have that C_j^m maps $c_{a_j}, c_{a_j+1}, \ldots, c_{b_j}$ onto $d_{a_j}, d_{a_j+1}, \ldots, d_{b_j}$. Let M_j be the set of such integers m. Then $E \models s = t$ iff all M_j are non-empty and have a common element, for the cycles M_j appearing in ϕ. It is not hard to show that if z_1 and z_2 are in M_j and $z_1 > z_2$ then $z_1 + c(z_1 - z_2)$ is also in M_j for all $c \geq 0$. This is because applying C_j z_1 or z_2 times has the same result, therefore applying C_j $z_1 - z_2$ times is the identity. It follows by a simple number theoretic argument that M_j is cyclic, that is, for some integers m_j and n_j, M_j contains exactly the integers of the form $m_j + c * n_j$ for all $c \geq 0$. One way to see this is to note that n_j is the smallest positive integer such that n_j applications of the cycle C_j map the string $(d_{a_j}, d_{a_j+1}, \ldots, d_{b_j})$ onto itself. These values m_j and $c * n_j$ can be obtained in polynomial time just by examining integers up to n, because they are bounded by $b_j - a_j$. Also, we can determine if the M_j for the cycles C_j in ϕ have a common element by an application of standard methods in number theory, and this can be done in polynomial time.

In particular, the equations $z \equiv m_j \pmod{n_j}$ have a common solution iff for all distinct i and j, $gcd(n_i, n_j)$ divides $m_i - -m_j$ ([NZ72], page 32). The greatest common divisor and the division test can be performed in polynomial time. But $z \equiv m_j \pmod{n_j}$ iff $z \in M_j$, hence we can determine in polynomial time whether the M_j have a common element.

We now show that this result can be extended to the case in which s and t may not be flat. Suppose s is of the form $f(s_1, \ldots, s_n)$ and t is $f(t_1, \ldots, t_n)$. As before, it suffices to consider the case in which s and t are ground terms. We want to determine whether $E \models s = t$ for a set E of equations containing a single flat permutation. For this, we recursively test whether $E \models s_i = t_j$ for all i and j. Using this information, we then partition the arguments s_i and t_j into equivalence classes, depending on whether they are E-equal. Then we can replace each equivalence class by a unique representative, obtaining terms s' and t' in which the arguments s_i and t_i have been replaced by constant symbols, each equivalence class being represented by a unique constant symbol. Then $E \models s = t$ iff $E \models s' = t'$. Since $s' = t'$ is a flat equation, this problem $E \models s' = t'$ can be solved in polynomial time, as shown above. The completeness of this algorithm follows from theorem 5.4. □

Corollary 5.13 *Suppose E is an equational theory consisting of flat permutative equations, each with a different function symbol. Then the equational theory of E can be decided in polynomial time.*

Proof. The algorithm is in fact exactly the same as in the theorem. □

We do not know if the result can be extended to a single leaf permutation.

We note that this result may involve very long proofs. The length of the minimal proof of $s = t$ from E may be exponential. However, this proof (or the existence of the proof) is decided in polynomial time. This shows an advantage of this permutation-based approach, namely, it can decide the existence of long proofs without actually constructing them. This is in contrast to the traditional term-rewriting approaches, where the proofs have to be constructed step by step. Of course, one might consider that E-unification algorithms for various theories E also shorten the proofs by making some steps implicit.

5.5 Matching and Flat Permutations

Theorem 5.14 *The problem of whether there exists a substitution Θ such that $E \models s_1 = (s_2\Theta)$, where E is a flat permutative theory and s_1 and s_2 are flat terms, is NP-complete, even for a theory E containing at most two equations.*

Proof. The construction is almost the same as for theorem 5.7. The set E of permutations is the same, and the encoding of the directed graphs H_1 and H_2 by the terms s_1 and s_2 is almost the same. The difference is that the edges not present in H_2 are encoded in the term s_2 by arguments c_i that are distinct new variables, rather than zeroes. Then a Θ as in the theorem exists iff H_2 is (isomorphic to) a subgraph of H_1, since these variables can be replaced arbitrarily by zeroes or ones and because E permits any permutation of the vertices to be simulated. However, the problem of whether one graph contains a subgraph isomorphic to another graph, is NP-complete. Membership of the given problem in NP follows from corollary 5.3. □

As before, this result depends on the fact that k has a large number of arguments, and this may not be realistic in practice. The question arises whether such a result can be preserved in the context of a bounded number of arguments.

Theorem 5.15 *The matching problem relative to a set of two leaf permutations is NP-complete. This is still true when the arity is bounded.*

Proof. As before in theorem 5.8, this is done by encoding the term $f(c_1, c_2, \ldots, c_{n^2})$ by the term $j(c_1, j(c_2, \ldots, j(c_{n^2-1}, c_{n^2}) \ldots))$. The rest of the argument is straightforward. However, we do not know whether this problem is in NP. □

5.6 Matching for Flat Permutations of Bounded Arity

We also are interested in the complexity of the matching problem for a group of flat permutations of bounded arity. The following result is actually a trivial consequence of the NP-completeness of commutative matching, shown in [BKN87]. However, we present a proof for purposes of comparison.

Theorem 5.16 *The problem of matching relative to a flat permutative theory E is NP-complete, even for bounded arities. This is true even for a theory E containing a single flat permutation.*

Proof. First we show this reslt for E containing several flat permutations, and then modify the result for E containing a single flat permutation. The idea is to construct a term s of the form $f(\ldots h(X_1, Y_1), \ldots, h(X_n, Y_n), \ldots, k(C_1, \_, \_), \ldots, k(C_n, \_, \_), \ldots)$ and a term t of the form $f(\ldots h(false, true), \ldots, h(false, true), \ldots k(u_1, u_2, u_3), \ldots, k(u_1, u_2, u_3), \ldots)$ where the $\ldots$ indicate extra term structure involving enough applications of functions of bounded arity to make a big enough term to have this many subterms. The $\_$ indicate distinct new variables and the u_i are terms defined below. The theory E involves the equation $h(x, y) = h(y, x)$; this means that in matching s with t, we choose independently whether X_i will be **true** and Y_i will be **false** or the reverse. The C_i encode three-literal clauses, that is, they are subterms of the form $k(Z_1, Z_2, Z_3)$ where the Z_i are chosen from the variables X_j and Y_j. In t we include many clauses having at least one of the three variables Z_i **true**, enough to permit a match whenever at least one of the Z_i are true. We define the term u_1 as $k(true, false, false)$, u_2 as $k(true, true, false)$, and u_3 as $k(true, true, true)$. The equational theory allows k to permute its arguments in all possible ways, and then we can have a choice whether to match the clause $k(Z_1, Z_2, Z_3)$ with $k(true, false, false)$, $k(true, true, false)$, or $k(true, true, true)$. Such a match is possible iff at least one of the Z_i is bound to **true**. We have a subterm of the form $k(k(true, false, false), k(true, true, false), k(true, true, true))$ in t at the same position in t as a subterm of the form $k(C_i, Y, Z)$ in s. Then a match of s with t corresponds to a satisfying truth assignment to a set of clauses encoded by s and t. Since this problem is NP-complete, the matching problem is NP-hard. To show membership in NP, we note that if the matching substitution Θ is exhibited, then we can determine if $E \models r_1 = r_2$ in nondeterministic polynomial time as shown in corollary 5.5. Thus this problem is NP-complete.

To obtain the result for a single flat permutation, we simulate the terms of the form $k(X, Y, Z)$ by a term of the form $h(h(X, Y), h(Z, W))$ where W is a new variable. Then we have to modify the structure of the terms $k(u_1, u_2, u_3)$ and the corresponding terms $k(C_i, \_, \_)$ to $h(h(u_1, u_2), h(u_3, u_4))$ and $h(h(C_i, \_), h(\_, \_))$, respectively, where u_4 is $h(h(true, false), h(true, false))$. Then if C_i is satisfied, the $true$ literal can be placed in front, and there are enough possibilities from the permutation $h(x, y) = h(y, x)$ to make the clause C_i match one of the u_i. □

Another interesting question is that of matching relative to a single leaf permutation; since flat permutations are leaf permutations, this is also NP-hard. However, we do not know if this problem is in NP.

5.7 Harder Questions

In [Fro89], the class of *symbol permutative* equations was also defined. These are equations that do not change the variables but only the interior symbols. That is, an equation $s = t$ is symbol permutative if each variable appears once in s and t and the sequence of variables is the same. Also, each function symbol appears the same number of times in s as in t. For example, the equation $f(x, f(y, z)) = f(f(x, y), z)$ is symbol permutative. The question arises as to the complexity of symbol permutative theories.

Theorem 5.17 *Suppose E is a symbol permutative equational theory. Then the decision problem for the equational theory of E is complete for polynomial space.*

Proof. One can see that the decision problem for such a theory is in polynomial space, since the sizes of the terms never grow in an equational derivation between s and t, and one can nondeterministically look for such a derivation. This uses the fact that nondeterministic polynomial space is equal to deterministic polynomial space. We can show that this problem is hard for polynomial space, by reduction from the acceptance problem for a linear bounded automaton. We can simulate the action of a linear bounded automaton on a term of the form $f_1(f_2(\ldots f_n(c)\ldots))$ where the sequence of f_i corresponds to the sequence of tape symbols in a linear bounded automaton, and one of the f_i encodes the state of the automaton. Then one can encode the moves of the automaton in a straightforward way in an equational theory E. The problem is that this encoding is not symbol permutative. However, by another encoding, this can be achieved. It suffices to encode each symbol f_i as a sequence $h^i k h^{m-i} k$ where h and k are unary function symbols and there are m different symbols altogether. With this encoding, every f_i is a permutation of every other f_j, so the theory E, so represented, is symbol permutative. It follows by this argument that the decision problem for symbol permutative equational theories is complete for polynomial space. $\square$

Of course, we can make the problem still harder by removing all restrictions and permitting E to have any form whatever. In this case, we can easily simulate a Turing machine using standard techniques and obtain problems that are at best partially decidable.

References

[BDP89] Leo Bachmair, N. Dershowitz, and D. Plaisted. Completion without failure. In Hassan Ait-Kaci and Maurice Nivat, editors, *Resolution of Equations in*

Algebraic Structures 2: Rewriting Techniques, pages 1–30, New York, 1989. Academic Press.

[BKN87] D. Benanav, D. Kapur, and P. Narendran. Complexity of matching problems. *Journal of Symbolic Computation*, 3(1,2):203–216, 1987.

[DJ90] N. Dershowitz and J.-P. Jouannaud. Rewrite systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science.* North-Holland, Amsterdam, 1990.

[Dom92] E. Domenjoud. Number of minimal unifiers of the equation $\alpha x_1 + ... + \alpha x_p =_{AC} \beta y_1 + ... + \beta y_q$. *Journal of Automated Reasoning*, 8:39–44, 1992.

[FHL80] M. Furst, J. Hopcroft, and E. Luks. Polynomial-time algorithms for permutation groups. In *Proceedings of the 21st IEEE Symposium on Foundations of Computer Science*, pages 36–41, Syracuse, New York, 1980.

[Fro89] I. Frobenius. Effiziente behandlung von äquivalenzklassen von termen. Technical Report Projektarbeit, Universität Kaiserslautern, Kaiserslautern, Germany, 1989.

[Klo92] Jan Willem Klop. Term rewriting systems. In S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 2, chapter 1, pages 1 – 117. Oxford University Press, Oxford, 1992.

[KN92] D. Kapur and P. Narendran. Double-exponential complexity of computing a complete set of AC-unifiers. In *Proceedings 7th IEEE Symposium on Logic in Computer Science*, pages 11–21, Santa Cruz, California, 1992.

[NZ72] I. Niven and H. Zuckerman. *An Introduction to the Theory of Numbers.* Wiley and Sons, New York, 1972.

[Pla93] D. Plaisted. Equational reasoning and term rewriting systems. In D. Gabbay, C. Hogger, J. A. Robinson, and J. Siekmann, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 1, pages 273–364. Oxford University Press, 1993.