Unification and Equation Solving in Nilpotent Groups and Monoids

Edmund Kieran Burke

Submitted in accordance with the requirements

for the degree of PhD.

The University of Leeds

School of Computer Studies

June 1991

Abstract

Unification and equation solving have been considered for groups [44], semigroups [43], abelian groups [39] and abelian semigroups [25],[33],[68],[69]. In this thesis we consider partially commutative groups and monoids. Nilpotency provides us with a partial commutativity condition in the case of groups.

It is noted that unification in a theory is equivalent to equation solving in a free model of that theory if such a model exists.

The unification algorithm for nilpotent groups G of class k works by passing to the quotient of G by the (k-1)th term of the lower central series and lifting the solution up the factors formed from G by the terms of the lower central series. There are certain unification problems, however, where this does not work as it stands and special treatment is required involving the solutions of a certain restricted class of diophantine equations of degree k.

The unification problem for the theories of nilpotent groups of class ≥ 5 is shown to be undecidable. This improves the result of Romankov [60] who showed it for class ≥ 9. The result is established by reducing the problem to that of algorithmically solving an arbitrary diophantine equation of degree 4. It is well known that this problem is undecidable [50], [60].

A special set of partially commutative monoids is introduced. An algorithm to solve equations in these monoids relative to solving certain systems of diophantine equations of degree 2 is given. These equations have similarities with those that occur for nilpotent groups of class 2.

Contents

Acknowlegments

I would like to thank my supervisor Dr.J.K.Truss for all the support and encouragement he has given me. I would also like to thank Dr.K.McEvoy and Mr.J.Derrick for their help and advice, especially in the early days of the project.

Chapter 1: Introduction

1.1 An overview of the thesis

In this overview we will take each chapter of the thesis in turn and give an outline of the contents. Before we do this, however, we will set out the aims of the project.

1.1.1 The aims of the project:

Unification has been considered for groups and semigroups. It has also been considered for abelian groups and abelian semigroups (see [25],[33],[39],[43],[44],[68],[69]). The main aim of this project is to study unification with respect to partially commutative groups and monoids. This partial commutativity is provided, in the case of groups, by the concept of nilpotency. We cannot use this concept when considering monoids but we study certain monoids which have 'nilpotent like' character.

1.1.2 Chapter 1

This chapter is an introduction to the thesis and a survey of the field of unification theory. Although unification has been considered in higher order theories it has been mainly concerned with the study of algorithmic methods for solving equations in first order theories (and their models). In this chapter the scene is set in 1.2 by introducing and defining first order languages, theories and structures. We then formally define the idea of unification and consider it within the context of logic programming (by considering resolution and paramodulation). We also introduce the concept of cardinality type.

1.3 is a survey of the work already carried out in the areas

that directly appertain to the research presented in this thesis.

Unification, as we define it, is a *syntactic* concept i.e. we define it for first order theories. It is, however, often easier to argue semantically. We show in 1.4 that this amounts to working in the free object, when it exists.

In 1.5 we look at examples of unification in some particular first order theories.

1.1.2 Chapter 2

In this chapter we consider unification for nilpotent groups. Section 2.1 is an introduction to the chapter.

In 2.2 we present certain preliminary definitions and results that will be required later on. We introduce the idea of basic commutators, so called because they form bases of the factors of successive terms of the lower central series of a nilpotent group. We outline a well known method, called the collecting process, which will algorithmically write a word in terms of these basic commutators. We also present a multiplicative commutator identity (Lemma 2.2.7) which is widely used throughout Chapters 2 and 3. Finally we finish this preliminary section off by constructing an algorithm which can test whether or not any word of the last non-trivial member of the lower central series of a nilpotent group is, or can be made by variable substitution, into an hth power (for any integer h). The algorithm determines all such substitutions if they exist.

Section 2.3 presents an algorithm for solving any equation in a free nilpotent group of class k relative to having an algorithm which will solve systems of kth power free diophantine equations of degree k. It is then shown that the theory of

nilpotent groups of class 2 is of nullary cardinality type.

Section 2.4 is a more detailed study of unification in nilpotent groups of class 2. We give a more accurate description of the square free diophantine equations of degree 2 that occur and label them τ-equations. In section 2.3 we showed that in general the theory of nilpotent groups of class 2 is nullary. We now consider a two-sorted theory of nilpotent groups of class 2 where the sorts pick out the elements of the commutator subgroup. In this two-sorted theory the cardinality type, in general, becomes dependent upon the cardinality type of systems of τ-equations rather than being nullary (it is in any case infinitary). We then briefly discuss work undertaken by Repin (see 1.3) which considers problems in just one variable.

Section 2.5 extends the idea introduced in 2.4 (for nilpotency class 2) of passing to a two-sorted theory to passing to a k-sorted theory for nilpotency class k.

We close this chapter by relating the work done here to work undertaken by U.Martin on special p-groups (see 1.3). Special p-groups are a certain kind of nilpotent group of class 2. We also extend these to higher nilpotency class.

1.1.4 Chapter 3

In this chapter we turn our attention to the consideration of the decidability of the unification problem in nilpotent groups. Section 3.1 is an introduction to the chapter.

In 3.2 we introduce a particular free nilpotent group of class 5 (which we call F). We also construct the six basic commutators (of weight 5) of F. The collecting process is then used to derive certain facts (Lemmas 3.2.2 and 3.2.3) about these basic commutators.

We begin 3.3 by recalling the Matiyasevitch result (which says that the problem of solving arbitrary diophantine equations is undecidable) in the form in which we require it. We then show that the unification problem for nilpotent groups of class 5 is undecidable by considering F and reducing the problem to the arbitrary diophatine equation problem introduced above. This result is then extended to nilpotency class > 5.

In 3.4 we close this chapter by briefly discussing the decidability of the unification problem when restricting to problems in one variable only. This one variable work was originally considered by Repin (see 1.3) although he considered equation solving in free nilpotent groups (it is shown in 1.4 that this is equivalent to the unification problem).

1.1.5 Chapter 4

In this chapter we turn our attention away from nilpotent groups toward partially commutative monoids. Section 2.1 is an introduction to the chapter.

In 2.2 the rather complicated systems of diophantine equations (which we call ρ-systems), that occur naturally later on, are introduced. The two classes of structures (which we call $\mathcal{N}$ and $\mathcal{M}$) that we will be considering are then presented.

In 4.3 we first show how to write words lying in a member of $\mathcal{N}$ in a certain canonical form. We then construct an algorithm to solve equations in any member of $\mathcal{N}$ relative to having an algorithm which will solve ρ-systems. We then present an example of the algorithm in action and show that there is no direct way of proving the equivalence of the problems of solving ρ-systems and solving equations in the members of $\mathcal{N}$.

In 4.4 we consider solving equations in the members of $\mathfrak{M}$. These structures are 'more' free than those in $\mathfrak{N}$ but equation solving in the members of $\mathfrak{M}$ is much more difficult than in the members of $\mathfrak{N}$ and these difficulties are discussed. It is shown that they do not arise in the 'smallest' elements of $\mathfrak{M}$, i.e. the two three-generator elements, but that they cannot be avoided when considering more than three generators.

In 4.5 we consider extending the work done in this chapter in similar directions to those taken in Chapter 2. We mention the extension of this work to partially commutative monoids that are less commutative and this corresponds to the idea of higher nilpotency class. We also consider the inclusion of the axiom $x^p = 1$, for some odd prime p. This turns the monoid into a group and as such has been considered in Chapter 2. We also consider cardinality type and problems in just one variable.

1.1.6 Chapter 5

This chapter concludes the thesis by providing a summary of the results obtained and presenting some directions for future research.

1.2 Preliminary definitions and notation:

We introduce here, not only the basic definitions and notations that we will be using throughout this thesis, but also some integral concepts that provide the background to the results presented in the subsequent chapters.

1.2.1 First order languages and theories

Let $\mathcal{L}$ be a first order language with equality as described in [23] or [51]. We will be using the standard definitions of $\mathcal{L}$-terms and $\mathcal{L}$-formulas which can again be found in [23] or [51]. We will say that an $\mathcal{L}$-formula ϕ is an *$\mathcal{L}$-sentence* if every variable of ϕ falls under the scope of a quantifier in ϕ. A set of $\mathcal{L}$-sentences closed under deduction will be called an *$\mathcal{L}$-theory*. An $\mathcal{L}$-theory is called an *equational theory* if the equality predicate is the only predicate which appears in the $\mathcal{L}$-sentences of the $\mathcal{L}$-theory and if it is axiomatized by universal sentences.

A *literal* is an atomic $\mathcal{L}$-formula or the negation of an atomic $\mathcal{L}$-formula. A *positive* literal is simply an atomic $\mathcal{L}$-formula and a *negative* literal is the negation of an atomic $\mathcal{L}$-formula. A *clause* is a disjunction of literals. A *Horn* clause is a clause with at most one positive literal. A *program* clause is a Horn clause with exactly one positive literal. A *goal* clause is a Horn clause with no positive literals. A *logic program* is a set of program clauses.

An *$\mathcal{L}$-structure* $\mathcal{A}$ is a quadruple $(A, Rel^{\mathcal{A}}, Fun^{\mathcal{A}}, Con^{\mathcal{A}})$ where:

(a) A is a non-empty set of objects which the variables of $\mathcal{L}$ range over.

(b) $Rel^{\mathcal{A}} = \{R^{\mathcal{A}} : R$ is a relation symbol of $\mathcal{L}$ and $R^{\mathcal{A}}$ is a relation on A which is an interpretation of R (having the correct arity)$\}$

(c) $Fun^{\mathcal{A}} = \{f^{\mathcal{A}} : f$ is a function symbol of $\mathcal{L}$ and $f^{\mathcal{A}}$ is a function on A which is an interpretation of f (having the correct arity)$\}$

(d) $Con^{\mathcal{A}} = \{c^{\mathcal{A}} : c$ is a constant symbol of $\mathcal{L}$ and $c^{\mathcal{A}}$ is a constant in A which is an interpretation of c$\}$.

Let $\mathcal{A}$ be an $\mathcal{L}$-structure and let ϕ be an $\mathcal{L}$-formula. We define ϕ being *true* in $\mathcal{A}$ (written $\mathcal{A} \models \phi$) in the standard manner (again see [23]or [51]). $\mathcal{A}$ is said to be a *model* of an $\mathcal{L}$-theory E if $\mathcal{A} \models \phi$ for every $\phi \in$ E. We denote the class of all models of E by Mod(E).

Now having introduced the idea of a model (of an $\mathcal{L}$-theory E) we wish to consider certain types of mappings between different models of E. We will be looking at isomorphisms and, in particular, homomorphisms. Let $\mathcal{A}$ and $\mathcal{B}$ be $\mathcal{L}$-sructures. A map $F: \mathcal{A} \to \mathcal{B}$ is said to be an *isomorphism* if it maps A 1-1 onto B and: (a) For each n-place relation symbol R of $\mathcal{L}$

$$\mathcal{A} \models R(a_1, \ldots, a_n) \Leftrightarrow \mathcal{B} \models R(F(a_1), \ldots, F(a_n))$$

for every $a_1, \ldots, a_n \in A$.

(b) For each n-place function symbol f of $\mathcal{L}$

$$F(f^{\mathcal{A}}(a_1,\ldots,a_n)) = f^{\mathcal{B}}(F(a_1),\ldots,F(a_n))$$

for every $(a_1,\ldots,a_n) \in A^n$.

(c) For each constant symbol c of $\mathcal{L}$

$$F(c^{\mathcal{A}}) = c^{\mathcal{B}}.$$

F is said to be a *homomorphism* if it maps A into B and (a) is replaced by

(a') For each n-place relation symbol R of $\mathcal{L}$

$$\mathcal{A} \models R(a_1,\ldots,a_n) \Rightarrow \mathcal{B} \models R(F(a_1),\ldots,F(a_n))$$

where $a_1,\ldots,a_n \in A$.

F is said to be an *endomorphism* if it is a homomorphism from $\mathcal{A}$ to $\mathcal{A}$.

As we will show in 1.4, the idea of a free model is very important when we wish to consider unification from a semantic point of view. We now introduce this concept. Let K be a set of $\mathcal{L}$-sructures and let $\mathcal{A} \in K$. $\mathcal{A}$ is said to be *free* in K if there is $X \subseteq A$ (called a *free basis*) such that for any $\mathcal{B} \in K$ and any map p: $X \to \mathcal{B}$, p extends to a unique homomorphism from $\mathcal{A}$ to $\mathcal{B}$. Suppose we have an $\mathcal{L}$-theory E, then $\mathcal{A}$ is said to be a *free model of E* if $\mathcal{A}$ is free in K = Mod(E).

Now let us consider the idea of unification in a particular $\mathcal{L}$-theory.

1.2.2 Unification

Let $\mathcal{L}$ be a first order language with equality and let E be an $\mathcal{L}$-theory. A *substitution* is a mapping from the set of variables of $\mathcal{L}$ to the set of terms of $\mathcal{L}$. Any substitution is equal to the identity mapping in all but a finite number of cases so it can be represented as a finite set $\{t_1/x_1,\ldots,t_n/x_n\}$ where the x_i are variables of $\mathcal{L}$ and the t_i are $\mathcal{L}$-terms.

Let V be a set of variables of $\mathcal{L}$. Two substitutions σ,θ are *equal over V*, $\sigma =_E \theta\ |V|$, if $\forall x \in V\ x\sigma =_E x\theta$ (substitutions will be written on the right).

We shall consider the following quasi-ordering on substitutions:

$$\sigma \geq_E \theta\ |V| \text{ if } \theta\lambda =_E \sigma\ |V| \text{ for some } \lambda$$

(from now on when the set of variables V is understood we will omit the $|V|$ from the notations $=_E\ |V|$ and $\leq_E\ |V|$).

This is not a partial ordering but can be made into one in the canonical way i.e. we factor out by the relation (on the set of all substitutions) which consists of the pairs of substitutions which are instances of each other in the theory E.

Suppose we have two $\mathcal{L}$-terms s,t. θ is said to *unify* s and t in E if $s\theta =_E t\theta$. We denote the set of all unifiers of s and t by $U_E(s,t)$. A *complete set of unifiers* of s,t is a set $cU_E(s,t)$ such that:

(1) $cU_E(s,t) \subseteq U_E(s,t)$.

(2) $\forall\theta \in U_E(s,t)\ \exists\sigma \in cU_E(s,t)\ (\ \theta \geq_E \sigma\ |\text{variables of } s,t|)$.

$cU_E(s,t)$ is called a *set of most general unifiers*, written $\mu cU_E(s,t)$ if and only if, in addition, we have:

(3) $\forall\theta,\sigma \in cU_E(s,t)\ \sigma \geq_E \theta \Rightarrow \sigma = \theta$

A *unification algorithm* for a theory E is an algorithm which takes as input an arbitrary pair s,t of $\mathcal{L}$-terms and generates a non-empty subset of $U_E(s,t)$, provided s,t are unifiable (and if not, it reports the fact).

The *Unification Problem* for a first order theory with equality E is, 'given any two terms, s and t, of E can we algorithmically find a substitution of the variables of s and t which will satisfy $s = t$ in E ?' i.e. the unification problem for E asks whether or not there is a unification algorithm for E.

A *minimal* unification algorithm is one which generates some $\mu U_E(s,t)$.

The cardinality of $\mu U_E(s,t)$ defines the following classes of $\mathcal{L}$-theory:

(a) An $\mathcal{L}$-theory is of *cardinality type unitary* if for any unifiable $\mathcal{L}$-terms s,t some $\mu U_E(s,t)$ exists and $|\mu U_E(s,t)| = 1$.

(b) An $\mathcal{L}$-theory is of *cardinality type finitary* if for any unifiable $\mathcal{L}$-terms s,t some $\mu U_E(s,t)$ exists and $|\mu U_E(s,t)|$ is finite.

(c) An $\mathcal{L}$-theory is of *cardinality type infinitary* if for any

unifiable $\mathcal{L}$-terms s,t some $\mu U_E(s,t)$ exists and there exist $\mathcal{L}$-terms s',t' such that $|\mu U_E(s',t')|$ is infinite.

(d) An $\mathcal{L}$-theory is of *cardinality type nullary* if it is not in (b) or (c).

We may use similar terms to describe the solution of equations (with arbitrarily many unknowns) in particular models of E (see 1.4). Thus we may say that an equation has a *most general solution* provided all other solutions are substitution instances of it. We may talk about a particular model of E being unitary if the set of most general solutions of any equation in the model has cardinality 1, and so on.

We have defined unification for a single pair of $\mathcal{L}$-terms s,t. Bückert, Herold and Schmidt-Schauss [7] have presented a specific $\mathcal{L}$-theory such that a set of most general unifiers exists for all problems of the form s = t ?, where s and t are unifiable, but such that a set of most general unifiers of a system $s_i = t_i$? $(1 \leq i \leq m)$ does not necessarily exist. For this reason unification problems are defined by means of systems in some places in the literature. Throughout this thesis we shall, however, consider unification for a single pair only. In Chapter 4 we are mainly concerned with showing that the problem of algorithmically solving equations in certain partially commutative monoids reduces to the problem of algorithmically solving certain special types of systems of diophantine equations which we will call "ρ-systems" (see 4.2.1). If we were to consider the problem of solving m equations simultaneously in one of our monoids then we could use exactly the same methods as in 4.3 to reduce the group-theoretic problem to the

number-theoretic problem of solving m ρ-systems simultaneously. From a number-theoretic point of view this may be more difficult but from a group-theoretic point of view the reduction is no more complicated than the single-equation case.

1.2.3 The Resolution Principle

Suppose t_1 and t_2 are $\mathcal{L}$-terms which are unifiable in the empty theory (i.e. the theory that does not have any non-logical axioms) and that θ is the most general unifier of t_1 and t_2 (the empty theory is unitary [58]). Suppose also that C_1 and C_2 are clauses.

The *resolution rule* says that from $C_1 \vee p(t_1)$ and $C_2 \vee \sim p(t_2)$ we are allowed to derive $C_1\theta \vee C_2\theta$ (where p is an atomic $\mathcal{L}$-formula).

The way most logic programming systems work is that they add a goal clause to a logic program, which is the negation of the formula to be proved, and then they attempt to derive a contradiction using resolution.

Suppose t_1, t_2 and t_3 are $\mathcal{L}$-terms where t_1 and t_2 are unifiable in the empty theory. Suppose that '=' is a symbol of $\mathcal{L}$ (i.e. '=' is the symbol that stands for equality) and that θ is the most general unifier of t_1 and t_2.

The *paramodulation rule* says that that from $C_1 \vee p(t_1)$ and $C_2 \vee$ '$t_2 = t_3$' we can derive $C_1\theta \vee C_2\theta \vee p\theta([t_3\theta])$ where $p\theta([t_3\theta])$ denotes the replacement of exactly one occurrence of

$t_1\theta$ by $t_3\theta$ in $p\theta$.

This rule arises as part of an attempt to design a logic programming system which can handle equality. It is used in conjunction with resolution.

1.3 Related work

One of the most important proof rules proposed in the field of automatic theorem proving was the resolution principle that we defined in 1.2.3 above. This was presented by J.A.Robinson [58] in 1965. The concept of unification is an integral part of this resolution principle. Robinson presented an algorithm which would generate a unique most general unifier of two first order terms if they were unifiable. This algorithm is for the empty theory. The resolution principle (and hence unification) has since proved to be of immense importance in logic programming.

The modern computer programming language PROLOG uses resolution and a unification algorithm for the empty theory. The unification algorithm normally used, however, omits the occur check (this is defined in 1.4.2) in an attempt to improve efficiency and hence makes PROLOG unsound. Another drawback of PROLOG is that it does not have full first order equality. The introduction of equality axioms to a logic program creates many implementation problems. If there are several function symbols appearing in the program then it is impractical to introduce the large number of axioms that would be required to represent the substitutivity of equality. Also the equality axioms tend to generate many useless clauses. For example let us consider the axioms of reflexivity and symmetry as clauses in a logic program i.e. we have (1) $x = x$ and (2) $x = y \Rightarrow y = x$. Now we can apply the substitution $\{x/y\}$ to clause (2) and resolve with clause (1). The result of this application of the resolution rule, however, is the clause $x = x$ i.e. the axiom of reflexivity that we already had.

There have been a number of attempts to design a logic

programming system which can handle equality. Robinson and Wos [59] in 1969 proposed the addition of the paramodulation rule (defined in 1.2.3 above) to resolution. Goguen and Meseguer [19] presented the language EQLOG which uses 'narrowing' to solve equations and term rewriting to handle functions ('narrowing' is the paramodulation rule restricted to constant terms). Malachi, Manna and Waldinger [45] designed the language TABLOG. This is a programming language based on first order logic with equality. It does not use resolution as an inference rule but is still heavily dependent upon unification. W.A.Kornfeld [37] introduced 'PROLOG with equality' which at first uses the normal unification algorithm for the empty theory. If two terms are not unified by this algorithm then the system switches to an alternative algorithm which attempts to prove the two terms equal. If this algorithm succeeds then it provides the variable substitutions.

In 1972 G.Plotkin [55] considered unification for the theory consisting solely of the associativity axiom (i.e. the theory of semigroups). The motivation behind this was that the problems created by the inclusion of certain axioms among the axioms of the logic program could be removed by omitting the problem-causing axioms and incorporating them into the unification algorithm. It was shown here that the theory of associativity is infinitary i.e. there exist problems in this theory which have an infinite number of most general unifiers (see Examples 1.5.5 and 1.5.6). In the mid seventies Siekmann [65] and Stickel [68],[69] independently produced unification algorithms for the theory of commutativity and for the theory of abelian semigroups which is the combination of the theories of associativity and commutativity. The algorithm for abelian

semigroups can be trivially turned into one for abelian monoids. These theories were shown to be finitary i.e. any unifiable problem in these theories has a finite number of most general unifiers. The late seventies and early eighties saw a massive increase in the number of special unification algorithms. Such algorithms were constructed for the theories of idempotence [56], and commutativity with idempotence [56] both of which are finitary. The theories of distributivity [71], distributivity with commutativity [70], distributivity with associativity [70], and distributivity with both associativity and commutativity [70] were all shown to be infinitary.

Many unification problems have arisen which have been concerned with group theory. This is partly because some of the originally famous algorithmic problems did arise in group theory. Examples are the word problem, the conjugacy problem and the isomorphism problem. Suppose we are given two words lying in a particular group. The word and conjugacy problems ask if there is an algorithm to decide whether or not the two words are equal or conjugate respectively. The isomorphism problem asks whether or not two group presentations give isomorphic groups. It is clear that these problems are essentially computational in nature. Note the difference between the word problem and the unification problem. The unification problem asks whether or not the words can be *made* equal by variable substitution. A substantial number of the early workers in this area were group theorists, not only because of the existence of these classical algorithmic problems, but also because group theory provided a suitable environment in which to set and solve computational problems. J.Neubuser in 1967 presented a paper [52] which is a survey of the work carried out in the area of computational

group theory up to that time. Although somewhat out of date, this survey provides a good idea of the many varied computational problems arising in group theory. This paper appears in a volume entitled 'Computational Problems in Abstract Algebra' edited by J.Leech [40]. Most of the papers appearing in this volume are concerned with groups. It was in this volume that the Knuth-Bendix completion procedure was first presented [36]. The construction of this procedure is now an important landmark in the area of theoretical computer science. Indeed it is one of the cornerstones of the field of term-rewriting. More recently, J.Cannon has devised a software package called Cayley [9] which is concerned solely with the solving of problems in groups.

We have already briefly discussed the early work in unification for semigroups and abelian semigroups. Makanin in 1977 [43] gave an algorithm to decide whether or not any given word equation has a solution (i.e. whether or not there exists a solution to a unification problem in the theory of semigroups). H.Abdulrab and J.Pecuchet [1] have, very recently, presented a brief survey of the results in this area in which they outline a simplified version. In this paper they use the pig-pug method first introduced by Lentin [41] in 1972. This method enumerates the set of most general unifiers which appear as labels of paths in a graph. For uncomplicated problems the pig-pug method works quite well. There are, however, some problems for which the pig-pug graph is infinite. In this case one has to resort to Makanin's algorithm. An algorithm to enumerate the set of most general unifiers, for any problem, was presented by J.Jaffar [29] in 1990. In 1983 Makanin [44] showed that the unification problem for the theory of groups was decidable. He uses his

algorithm for semigroups, mentioned above, to achieve this.

J.M.Hullot in 1980 [26] provided a minimal unification algorithm for the theory of quasi-groups and showed that this theory is finitary. In 1984 Lankford, Butler and Brady [39] presented a unification algorithm for the theory of abelian groups. Repin [57] in 1985 constructed an equation solving algorithm (restricted to problems with just one variable) for free nilpotent groups of class 2. Repin also shows in the same paper that such an algorithm does not exist for free nilpotent groups of class c, where $c>10^{20}$. This immediately implies that (when restricting to one variable) there is a unification algorithm for the theory of nilpotent groups of class 2 and that the unification problem is undecidable for nilpotent groups of class c, where $c>10^{20}$ (see Theorem 1.4.2). The results presented in Chapter 2 study the extension of this to problems containing any number of variables. In Chapter 2 we also consider nilpotent groups of class k for any integer k. V.A. Romankov [60] in 1977 showed that the endomorphic reducibility problem is unsolvable for free nilpotent groups of class ≥ 9. This again immediately implies that the unification problem for such groups is undecidable. A result presented in Chapter 3 reduces this nilpotency undecidability boundary to class 5. Romankov [61] also showed in 1979 that the problem of algorithmically solving arbitrary equations in a free metabelian group is undecidable. Hence the unification problem for the theory of metabelian groups is undecidable. The extension of this result to groups of higher solubility class is trivial. Herold and Siekmann [25] extended the abelian semigroup results to allow unification in abelian semigroups with uninterpreted function symbols (we will discuss the theory of abelian groups with and without

uninterpreted function symbols in 1.5). We consider nilpotent monoids in Chapter 4. U.Martin and T.Nipkow [48] showed in 1986 that the theory of boolean rings is unitary i.e. any unifiable problem in this theory has a unique most general unifier. U.Martin has recently constructed a unification algorithm for special p-groups [47]. The relationship between this work and the work on nilpotent groups of class 2 is discussed in Chapter 2.

In 1966 W.F.Gould [20] showed that there are some problems in higher order theories where there exists an infinitely descending sequence of unifiers i.e. there is a unifiable problem, such that for any unifier θ there is another unifier θ' such that $\theta' < \theta$. Theories with this property are called nullary theories. The problem of whether or not first order nullary theories exist was an open problem for quite some time. It was not until 1983 that Fages and Huet [17] solved this by constructing a specific first order theory (i.e. the theory comprising of the axioms `1.x = x' and `f(x.y) = f(y)') which they showed to be nullary. In 1986 F.Baader [3] showed that the theory of idempotent semigroups is nullary. We will show in Chapter 2 that a single-sorted theory of nilpotent groups of class k is another first order theory which is nullary. We will also propose a method for getting round this problem by passing to a many-sorted theory.

There are many other areas of unification theory that are attracting considerable research activity. Unification and its applications in many-sorted logics has been considered by C.Walther [72], A.G.Cohn [12] and M.Schmidt-Schauss [63].

There has been much work recently on the combination of unification algorithms. Suppose we have a unification algorithm

for the theory E_1 and one for the theory E_2. The problem posed in this field is to determine if we can merge these algorithms together to produce a unification algorithm for the theory $E_1 + E_2$. Work in this area has been undertaken by K.Yellick [73], C. Kirchner [33], A.Herold [24] and M.Schmidt-Schauss [64].

A blossoming research area at the moment is that of constraint logic programming. It has already been mentioned above that certain axioms can be taken out of the logic program and grafted into the unification algorithm as long as we can solve the unification problem for the theory under consideration. However, unification theory has only been concerned with theories that contain just one predicate (i.e. the equality predicate). Constraint solving goes one step further and allows us to introduce inequalities as well as other specialised predicates. The set of axioms involving these other predicates, as well as equality, is known as a set of constraints. In a constraint logic programming system Robinson's unification algorithm for the empty theory is replaced, in the resolution step, by an algorithm to solve a system of constraints. Examples of current constraint logic programming systems are CHIP, which was developed by M.Dincbas et al [16], PROLOG II [13] and PROLOG III [14], which were both developed by A.Colmerauer. J.Jaffar and J-L.Lassez in 1987 [30] gave a theoretical treatment of logic programming with constraints. C.Kirchner, H.Kirchner and M.Rusinowitch [34] have considered constrained deduction for equational logic and for first order logic with equality. J.Cohen [11] has recently presented a survey of the work carried out so far in the field of constraint

logic programming. However, constraint solving is not concerned entirely with logic programming. Some research is currently taking place to solve more general problems using constraint solving methods. R.Feldman and M.C.Golumbic [18] have presented optimization algorithms for the student scheduling problem using constraint solving.

Many problems in unification theory reduce to the problem of algorithmically finding solutions for diophantine equations. The work presented in this thesis gives several examples of such problems. Lankford, Butler and Brady's algorithm [39] for abelian groups requires an algorithm to compute a basis for the solution space of systems of homogeneous linear diophantine equations as well as an algorithm to give a solution for arbitrary systems of linear diophantine equations. The unification algorithms for abelian semigroups and for abelian semigroups with uninterpreted function symbols ([68],[69],[25]) also rely heavily upon the algorithmic solutions of linear diophantine equations (see [28] or [35] for a presentation of these algorithms). In 1970 Matiyasevitch [50] showed that the problem of the existence of integer solutions for an arbitrary diophantine equation is undecidable (solution to Hilbert's tenth problem). The undecidability results of Romankov [60],[61] and Chapter 3 rest upon this result.

Apart from the uses already mentioned, in logic programming and equational logic programming, unification theory has other applications in many varied areas of computer science. Unification algorithms are used in programming languages, term-rewriting, natural language processing and in modal and temporal logics which are gaining importance in the field of artificial intelligence. In particular unification in

commutative theories has important uses which crop up in several different fields of computer science. A unification algorithm for abelian semigroups is used in term rewriting and in a number of different programming languages. As has been mentioned, the main motivation for the research presented in this thesis is that it may prove very useful to have unification algorithms for partially commutative theories. Nilpotency can be viewed as a partial commutativity condition. The lower the nilpotency class of the theory the more commutative is that theory. For example, nilpotent groups of class 1 are abelian groups.

1.4 Unification and free models

Unification as we have defined it is a syntactic notion. As can be seen in 1.2.2 it is defined for first order theories with equality. In many of the results presented in this thesis we prefer, however, to argue semantically i.e. in a model of the theory. The result of this section allows us to do so in cases where the appropriate free models exist. However, this result is of no use if the theory under consideration does not posess free models. We know [5] that a theory E has a free model if it is an equational theory and the theory of nilpotent groups of class k is equational. We shall show, however, in 4.5 that a free model does not exist for the theory of nilpotent monoids. Hence in the case of nilpotent monoids we cannot call upon the result of this section.

1.4.1 Definitions:

Let $\mathcal{L}$ be a first order language with equality, let E be an $\mathcal{L}$-theory and let $\mathcal{A}$ be a free model of E on the set $X = \{x_1, x_2, \ldots\}$ of free generators, where X is a countably infinite set of distinct elements which are going to be thought of as variables ranging over A. Let $T_{\mathcal{L}}$ be the set of $\mathcal{L}$-terms. We now define a map $I^{\mathcal{A}}: T_{\mathcal{L}} \to A$ inductively on $t \in T_{\mathcal{L}}$ as follows:

If t is a variable v_i let $I^{\mathcal{A}}(v_i) = x_i \in X$.

If t is a constant symbol c let $I^{\mathcal{A}}(c) = c^{\mathcal{A}}$.

If $t = f(t_1, .., t_n)$ where f is an n-place function symbol and $t_1, .., t_n$ are $\mathcal{L}$-terms then $I^{\mathcal{A}}(t) = f^{\mathcal{A}}(I^{\mathcal{A}}(t_1), .., I^{\mathcal{A}}(t_n))$.

Now consider the unification problem $t_1 = t_2$? in E. The following theorem allows us to work in $\mathcal{A}$ rather than in E:

Theorem 1.4.2:

Suppose a substitution θ and unification problem $t_1 = t_2$? are given. Then:

$$t_1\theta =_E t_2\theta \Leftrightarrow I^{\mathcal{A}}(t_1\theta) = I^{\mathcal{A}}(t_2\theta)$$

Proof:

From left to right the implication is obvious since $\mathcal{A}$ is a model of E. This is "soundness" (see [23] or [51]).

Conversely assume that $I^{\mathcal{A}}(t_1\theta) = I^{\mathcal{A}}(t_2\theta)$. Then by the Gödel Completeness Theorem (see [23] or [51]) it suffices to show that $\mathfrak{B} \models t_1\theta = t_2\theta$ for any $\mathfrak{B} \in$ Mod(E) and assignment to members of B of any free variables occurring. Let u be just such an assignment.

Now let $p:X \rightarrow \mathfrak{B}$ be defined by $p(x_i) = u(v_i)$. We know that because $\mathcal{A}$ is free p can be extended to a unique homomorphism $h:\mathcal{A} \rightarrow \mathfrak{B}$.

Since h is a homomorphism we have that $h(I^{\mathcal{A}}(t_1\theta)) = h(I^{\mathcal{A}}(t_2\theta))$.

Now we show that $h(I^{\mathcal{A}}(t)) = t^{\mathfrak{B}}$ where $t \in T_L$ by induction on the construction of t.

If t is a variable v_i then $I^{\mathcal{A}}(v_i) = x_i \in X$ and $h(x_i) = p(x_i) = u(v_i) = t^{\mathfrak{B}}$.

If t is a constant symbol c then $I^{\mathcal{A}}(c) = c^{\mathcal{A}}$ and because h is a homomorphism $h(c^{\mathcal{A}}) = c^{\mathfrak{B}} = t^{\mathfrak{B}}$.

If $t = f(t_1,..,t_n)$ where f is an n place function symbol and $t_1,..,t_n$ are members of T_L then $I^{\mathcal{A}}(t) = f^{\mathcal{A}}(I^{\mathcal{A}}(t_1),..,I^{\mathcal{A}}(t_n))$ and we have that

$h(I^{\mathcal{A}}(t)) = f^{\mathcal{B}}(h(I^{\mathcal{A}}(t_1)),..,h(I^{\mathcal{A}}(t_n)))$

(because h is a homomorphism)

$= f^{\mathcal{B}}(t_1^{\mathcal{B}},..,t_n^{\mathcal{B}})$ by induction

$= t^{\mathcal{B}}$.

Therefore we have $t_1\theta^{\mathcal{B}} = t_2\theta^{\mathcal{B}}$ in $\mathcal{B}$ for any assignment of the free variables occurring in $t_1\theta, t_2\theta$ to elements of B

i.e. $\mathcal{B} \models t_1\theta = t_2\theta$.

1.5 Some unification results and examples

We now look at some established results, together with examples, in order to introduce some of the concepts that we will be taking as read later on. The most widely used of all unification algorithms is the algorithm for the empty theory which we will denote by Ø. This was the first ever unification algorithm.

Theorem 1.5.1 (c.f [10] or [42]):

There is a minimal unification algorithm for Ø.

Moreover, Ø is unitary.

Remark 1.5.2:

Suppose a unification problem $t_1 = t_2$? in Ø is given. The algorithm of Theorem 1.5.1 works by attempting to unify each subterm of t_1 and t_2 from left to right. If upon attempting to unify two subterms one subterm is a variable x and the other a term t then we apply the substitution t/x to the problem. We can only do this, however, if the variable x does not occur in the term t. For the unification algorithm to work a check, which is called the *occur check*, must be carried out to ensure that this is the case. The necessity of the occur check can be seen by considering the problem f(x) = x? in Ø where x is a variable and f is one-place function symbol. Clearly there is no solution, but without the occur check the unification algorithm of Theorem 1.5.1 would carry on substituting f(x) for x infinitely many times.

The algorithm of Theorem 1.5.1 is non-deterministic in the sense that there may be more than one possible substitution to

make when the attempt is made to unify subterms. Different substitutions will, however, produce the same most general unifier up to variable renaming. This highlights the remark made in 1.2.2 that $\leq_E$ (in this case E is Ø) is a quasi-ordering which can be made into a partial ordering in the canonical way.

Definitions 1.5.3:

Let $\mathcal{L}$ be a first order language consisting of just one two-place function symbol f. We will use the following notations:

(a) C denotes the theory consisting of the commutativity axiom only i.e. $\forall x \forall y (f(x,y) = f(y,x))$.

(b) A denotes the theory consisting of the associativity axiom only i.e. $\forall x \forall y \forall z (f(f(x,y),z) = f(x,f(y,z)))$.

Note: When this holds we can consistently write xy for f(x,y) and there is no need to insert brackets.

(c) C+A denotes the theory obtained by combining both of the above axioms (for the same f).

The following theorem gives us results about the existence of minimal unification algorithms and the cardinality type of the theories defined in 1.5.3.

Theorem 1.5.4 ([55],[65],[68],[69]):

(a) There are minimal unification algorithms for each of C and C+A, and both theories are finitary.

(b) A is infinitary.

Let us now look in some detail at the theories we have introduced:

Example 1.5.5:

Suppose the unification problem f(x,a) = f(a,x) ? is given where a is a constant and x is a variable of the language under consideration. We consider this problem in each of the theories introduced above:

1.5.5.1: Let us first consider this problem in Ø. Here the only possible solution is θ = {a/x} and this is *the* most general unifier.

1.5.5.2: We now consider the problem in C. In this case we can substitute any term for x and make use of the commutativity axiom.

So θ is a unifier as it was for 1.5.5.1 but here σ = {b/x} (where b is a constant of $\mathcal{L}$) is also a unifier (note that σ is not a unifier in 1.5.5.1). However, σ is not a most general unifier since τ = {t/x} (where t is any term of $\mathcal{L}$) is also a unifier, $\tau \leq_C \sigma$ and $\tau \neq_C \sigma$ (even up to equivalence in C). In fact τ is the most general unifier for this problem in C. Note that this is a special case of a problem in C where we have one most general unifier. In general we have finitely many most general unifiers.

1.5.5.3: We now consider the problem in A. As for 1.5.5.1 we have that θ is a unifier. However, the substitutions σ,τ that were unifiers in 1.5.5.2 are not unifiers here. Clearly in this

case the set of substitutions $\{a^n/x: n \in \mathbf{N}\}$ forms the set of all unifiers of the problem in A. This is an infinite set of constant values and thus illustrates the infinitary nature of A.

1.5.5.4: In C+A for this particular problem we have exactly the same situation that we had in 1.5.5.2.

We will now consider a more involved problem:

Example 1.5.6:

Suppose we have the unification problem

$$f(x,f(a,f(b,y))) = f(y,f(b,f(a,x))) \ ?$$

where a,b are constants and x,y are variables of the language under consideration. We can simplify the notation by writing the problem in the form

$$x(a(by)) = y(b(ax)).$$

We will again consider the given problem in each of the theories introduced above:

1.5.6.1: We will first consider the problem in Ø. We can see that although we can unify the first two subterms from the left x,y we cannot unify the second two, the constants a,b. Thus there cannot be a solution of the problem in Ø.

We will now show that in C the problem is also insoluble.

1.5.6.2: If we look at the first subterm on the left hand side of the equation it can be seen that we have the choice of attempting to unify x with y or (by use of the commutativity axiom) x with b(ax). It is clear that we cannot unify x and b(ax). Therefore, if there is a solution to the problem we must have x = y. We are then left with the problem a(bx) = b(ax). As

a,b are constants they will not unify so (by use of the commutativity axiom again) we have to unify ax with a which is impossible (without the presence of an identity). Hence the problem has no solution in C. We will now show that not only do we get a solution of the problem in A but we get infinitely many solutions.

1.5.6.3: The problem can be rewritten as xaby = ybax ? because we are working with the associtivity axiom. By considering this problem we show that A is infinitary. The simpler problem of 1.5.5 shows this but we will still consider problem 1.5.6 in some detail to provide a non-trivial infinitary problem and to provide a clearer exposition of the concepts involved.

Claim 1:

The solutions to this are given by $\theta_n = \{a^n/x, a^{n+1}/y\}$, $(n \geq 0)$, $\varphi_n = \{b^{n+1}/x, b^n/y\}$, $(n \geq 0)$, or if we already have a solution $\{u/x, v/y\}$ then $\psi = \{u/x, uabv/y\}$ or $\eta = \{vbau/x, v/y\}$.

Proof:

It is clear that $x \neq y$ so let us assume first that the length of x is less than the length of y. We can see that y = xa or y = xabv for some (possibly empty) v.

If y = xa then we obtain xabxa = xabax. Thus xa = ax which means that $x = a^n$ for $n \geq 0$, so the solution is θ_n. Now if y = xabv then substituting this into the equation gives xabxabv = xabvbax i.e. xabv = vbax and $\{x/x, v/y\}$ is also a solution. Thus the solution is of the form of ψ. φ_n and η can be derived by assuming that the length of y is less than the length of x. Hence Claim 1 is proven.

Claim 2:

There cannot exist a solution in which a^2b^2 is a subword of x or y.

Proof:

We use induction on the construction of x and y given by Claim 1.

For the base case x and y are given by θ_n or φ_n.

For the induction step x and y are given by

(a) ψ = {u/x,uabv/y} for some solution {u/x,v/y}

or (b) η = {vbau/x,v/y} for some solution {u/x,v/y}.

We will consider cases (a) and (b) in turn:

(a) By the induction hypothesis a^2b^2 does not occur in u or v so the only way in which a^2b^2 can occur in y is if u ends with a and v begins with b.

Suppose by way of contradiction that u = u'a and v = bv' for some u' and v'.

$\therefore \psi$ = {u'a/x,u'a^2b^2v'/y}

Substituting into the original problem we have

u'a^2bu'a^2b^2v' = u'a^2b^2v'bau'a

$\therefore$ u'a^2b^2v' = bv'bau'a

$\therefore$ u' = bu'' and v' = v''a for some u'' and v''

$\therefore$ u = bu''a and v = bv''a

Now {u/x,v/y} is a solution to the problem so we have

bu''a^2b^2v''a = bv''ababu''a

The only way a^2b^2 can occur on the right hand side of the equation is if it occurs in u'' or v'' which contradicts the induction hypothesis.

(b) By the induction hypothesis a^2b^2 does not occur in u or v so the only way in which a^2b^2 can occur in x is if u begins with

a^2b or v ends with ab^2. Having noted this the proof follows in a very similar manner to that for (a).

Hence Claim 2 is proven.

Now assume A is finitary. Then we have a finite number of most general unifiers. However, each most general unifier must be constant since if a variable occurred we could substitute a^2b^2 for it which we cannot do by Claim 2. Therefore we only have finitely many solutions which contradicts Claim 1.

Now because every solution is constant they must all be most general. Thus A is infinitary.

1.5.6.4: We now consider the problem in A+C. We can rewrite it as xyab = xyab ? and the most general solution in this case is trivially {x/x,y/y}. Although we have not given an example which shows the finitary nature of A+C the example we give for the theory of abelian monoids (see 1.5.13), which is very similar to A+C, provides this. The theory of abelian monoids is A+C together with an identity. However, for the moment let us consider the theory of abelian groups which we will denote by AG. This consists of the group axioms + C.

Theorem 1.5.7 ([39]):

There is a minimal unification algorithm for the theory of abelian groups (with free constants).

Example 1.5.8:

Suppose we have the problem $x^3y^2z^{-1}a^2b^{-3}c^{-1} = 1$? in AG where x,y,z are variables and a,b,c are constants. By Theorem 1.4.2 this is equivalent to solving the equation in a free abelian

group of countably infinite rank. We can express the given problem in additive notation to obtain

$$3x + 2y - z + 2a - 3b - c = 0.$$

The algorithm of Theorem 1.5.7 makes use of an algorithm (see [28]) which generates a basis of the solution space of the homogeneous equation $3x + 2y - z = 0$:

$$R = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 3 \end{bmatrix}$$

(writing the solutions as columns).

We now use an algorithm (see [35]) to generate a particular solution:

$$S = \begin{bmatrix} b \\ -a \\ -c \end{bmatrix}$$

Suppose $Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ where y_1, y_2 are new variables and $V = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

then a most general unifier (we will show that it is *the* most general unifier in the next theorem) θ is given by $\{(RY + S)/V\}$ i.e. rewriting to multiplicative notation

$$\theta = \{y_2 b/x,\ y_1 a^{-1}/y,\ y_1^2 y_2^3 c^{-1}/z\}$$

We now show that this is the only most general unifier (up to the relation obtained in the canonical way that was mentioned in 1.2.2).

Theorem 1.5.9:

AG is unitary.

Proof:

Suppose the problem $x_1q_1 \ldots x_nq_nc_1p_1 \ldots c_mp_m = 1$ in AG is given.

Written additively this becomes

$$q_1x_1 + \ldots + q_nx_n + p_1c_1 + \ldots + p_mc_m = 0$$

where $x_1, \ldots, x_n$ are variables, $c_1, \ldots, c_m$ are constants and $q_1, \ldots, q_n, p_1, \ldots, p_m$ are integers. Now there is an algorithm [28] which computes $R = (r_1\ r_2\ \ldots\ r_s)$ where $\{r_1, r_2, \ldots, r_s\}$ forms a basis for the set of solutions of $q_1x_1 + \ldots + q_nx_n = 0$. A particular solution P of

$$q_1x_1 + \ldots + q_nx_n + p_1c_1 + \ldots + p_mc_m = 0$$

can also be computed [35]. The unifier given by Theorem 1.4.6 is

then $\theta = \{(RY + P)/V\}$ where $Y = \begin{bmatrix} y_1 \\ \cdot \\ \cdot \\ \cdot \\ y_s \end{bmatrix}$ and $V = \begin{bmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{bmatrix}$

($y_1, \ldots y_s$ are new variables).

In an attempt to simplify matters we will let

$$\theta = \{\theta_1/x_1, \ldots, \theta_n/x_n\}.$$

Suppose $\sigma = \{\sigma_1/x_1, \ldots, \sigma_n/x_n\}$ is any solution of the problem. We must show that there exists a substitution λ such that $\theta\lambda = \sigma$. Now applying the substitutions θ, σ to the problem and subtracting yields $q_1(\sigma_1 - \theta_1) + \ldots + q_n(\sigma_n - \theta_n) = 0$.

Therefore there exist integers

$\lambda_1,\ldots,\lambda_s$ such that $\begin{bmatrix}\sigma_1 - \theta_1\\ \sigma_2 - \theta_2\\ \cdot\\ \cdot\\ \sigma_n - \theta_n\end{bmatrix} = \lambda_1 r_1 + \lambda_2 r_2 + \ldots + \lambda_s r_s$

and so $\sigma = \theta + \lambda_1 r_1 + \lambda_2 r_2 + \ldots + \lambda_s r_s$

$$= \{\ R \begin{bmatrix}y_1 + \lambda_1\\ \cdot\\ \cdot\\ \cdot\\ y_s + \lambda_s\end{bmatrix} + P \begin{bmatrix}x_1\\ \cdot\\ \cdot\\ \cdot\\ x_n\end{bmatrix} \}.$$

If we let $\lambda = \{(y_1 + \lambda_1)/y_1, \ldots, (y_s + \lambda_s)/y_s\}$ then $\theta\lambda = \sigma$, as required.

Remark 1.5.10:

Lankford, Butler and Brady showed in [39] that the theory of abelian groups is finitary. They are, however, considering the theory of abelian groups with *uninterpreted function symbols* (which we will denote by AG'). As it is defined above AG has only the group operation function symbol. The following example will highlight the difference between the two and illustrate the finitary nature of AG'.

Example 1.5.11:

First of all, suppose the problem xy = ab? is given in AG where x,y are variables and a,b are constant symbols. Then the most general solution is $\theta = \{tab/x, t^{-1}/y\}$.

Now, suppose the problem f(x)f(y) = f(a)f(b)? is given in AG' where f is a one place uninterpreted function symbol. This problem is not unified by θ. It is, however, unified by

$$\sigma = \{a/x, b/y\} \text{ and } \tau = \{b/x, a/y\}.$$

Clearly σ,τ are not instances of each other in AG' (but they are instances of θ in AG).

We will now close this introductory chapter with a brief look at the theory of abelian monoids which we will denote by AM. This is A+C together with an identity. We have the following theorem.

Theorem 1.5.12 [25]:

There is a minimal unification algorithm for AM.

Moreover AM is finitary.

Example 1.5.13:

Suppose we consider the problem of Example 1.5.8 in AM i.e. we consider the problem $x^3y^2a^2 = z^1b^3c^1$? in AM where x,y,z are variables and a,b,c are constants. We can express the given problem in additive notation to obtain

$$3x + 2y - z + 2a - 3b - c = 0.$$

We now use the algorithm of [28] which generates a basis of the solutions of the homogeneous equation over the positive integers:

$$R = \begin{bmatrix} 0 & 1 \\ 1 & 0 \\ 2 & 3 \end{bmatrix}$$

As auxiliary problems the three equations:

$$3x + 2y - z = -2 \text{ for a.....(1)}$$

$$3x + 2y - z = 3 \text{ for b.....(2)}$$

$$3x + 2y - z = 1 \text{ for c.....(3)}$$

arise. There is an algorithm [25] for generating a basis of the solutions of (1),(2),(3) over the positive integers:

For (1) this yields {(0,0,2)},

for (2) {(1,0,0) (0,2,1)}

and for (3) {(0,1,1),(1,0,2)}.

The product of these three bases is:

{((0,0,2),(1,0,0),(0,1,1)) , ((0,0,2),(1,0,0),(1,0,2)) ,

((0,0,2),(0,2,1),(0,1,1)) , ((0,0,2),(0,2,1),(1,0,2))},

and this provides a complete minimal set of particular solutions. For instance from the first element of the product we can derive the matrix

$$P_1 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

and similarly we can derive the matrices P_2, P_3 and P_4 corresponding to the other elements of the product.

Let $U = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$, $A = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ and $V = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

then for i =1,..,4 the most general unifiers are:

$$\sigma_i = \{RU + P_i A/V\}$$

It is easy to check

(1) that none of these are instances of each other,

(2) that in AG each is an instance of the most general unifier generated in Example 1.5.8, and

(3) any unifier is an instance of a member of this set.

Now having introduced the basic ideas and concepts of Unification Theory and having looked at some of the established results we can begin to consider the main aim of this research project which is to investigate unification and equation solving in partially commutative theories and structures. We start Chapter 2 by considering nilpotent groups.

Chapter 2: Unification in nilpotent groups of class k

2.1 Introduction:

As we have seen in Chapter 1 unification algorithms have been constructed for various equational theories. Of particular interest are varieties of groups and semigroups. We have had a brief look at abelian groups. The case of groups was discussed in [44]. We study here an intermediate case, that of nilpotent groups of class k, some fixed k. We remark that in contrast with the two instances just cited the results presented in Chapter 3 show that unification for nilpotent groups of class k is undecidable for $k \geq 5$. Repin [57] showed that the problem of solving equations in just one variable is undecidable for $k > 10^{20}$. For $k = 2$ in the same paper he gave an algorithm to solve equations in just one variable (by Theorem 1.4.2 this is a unification algorithm for $k = 2$, for problems in just one variable). Here we extend this by giving relative unification algorithms, for any k and any number of variables, based on lifting Lankford, Butler, and Brady's algorithm [39] up the factors formed from G by the terms of the lower central series. The algorithms are relative to solving kth power free diophantine equations of degree k.

Now as we remarked earlier, unification, as we have defined it, is a syntactic concept. This is, however, one of the many cases where it is generally easier to argue semantically, and by Theorem 1.4.2 this amounts to solving equations in an appropriate *free object* (in cases where this exists), here a free nilpotent group of class k and infinite rank. When $k = 2$ we have a "nullary" theory, i.e. there are certain pairs of unifiable terms

which do not possess a most general unifier (it seems likely, but has not been proven, that we also have a nullary theory for k > 2). To attempt to correct this we pass to a many-sorted theory where the sorts pick out the terms of the lower central series.

A related and interesting problem is that of the solution of equations in groups where we think of the problem semantically from the word go. Unification then amounts to a version of this problem where the particular group is free nilpotent of class k and infinite rank. The methods also apply to solutions of equations in arbitrary free nilpotent groups of class k (of finite or infinite rank). We shall formulate the results for these and the unification results will follow automatically.

It has already been mentioned in 1.3 that related work has been carried out by Martin [47], who considered unification in special p-groups and Repin [57] who considered equation solving in free nilpotent groups of class 2 restricted to problems in just one variable. We will discuss the relationship between their work and the work presented in this chapter.

2.2 Preliminaries:

We now recall the definitions and basic results that we shall need:

Definition 2.2.1:

The element $x^{-1}y^{-1}xy$ of a group G is called the *commutator* of x and y, written [x,y]. We can define commutators of higher order by $[x_1, \ldots, x_n] = [[x_1, \ldots, x_{n-1}], x_n]$. We denote the element $y^{-1}xy$ by x^y.

The following lemma follows immediately from these definitions:

Lemma 2.2.2:

If x,y,z are elements of a group then:

(i) $[x,yz] = [x,z][x,y]^z$

(ii) $[xy,z] = [x,z]^y[y,z]$

(iii) $[x,y]^z = [z,[x,y]^{-1}][x,y]$

Definition 2.2.3:

Suppose we have a group G generated by $a_0, \ldots, a_k$. These are the *basic commutators of weight 1*, i.e. $wt(a_i) = 1$ for $i = 0, \ldots, k$. We order these by $a_i \leq a_j$ if $i \leq j$. If c and d are commutators of G then [c,d] is a commutator *of weight* wt(c) + wt(d). We order these by d < c if wt(d) < wt(c). If two commutators of weight > 1 have equal weight then we order them arbitrarily. The *basic commutators of weight n* are then the commutators [c,d] such that:

(1) c and d are basic

(2) wt(c) + wt(d) = n

(3) c > d

(4) If c = [e,f] then d ≥ f.

A group G is *nilpotent of class k* if there exists a normal series $G = G_0 \geq G_1 \geq \ldots \geq G_{k-1} \geq G_k = 1$ such that G_{i-1}/G_i lies in the centre of G/G_i for $i = 1,2,\ldots,k$. If such a series exists then G_i can be taken to be $[G_{i-1},G]$. This is called the *lower central series*. We will denote the *theory* of nilpotent groups of class k by N_kG.

Lemma 2.2.4 (c.f.[21] or [53]):

The basic commutators of weight k of a free nilpotent group of class k form a basis for the free abelian group G_{k-1}.

Lemma 2.2.5 ([21]):

Suppose c and d are commutators in a nilpotent group of class k. Let $d_0 = d$ and $d_{n+1} = [d_n,c]$.

Then (1) $dc = cd[d,c]$

(2) $d^{-1}c = c[d,c]^{-1}d^{-1}$

(3) $dc^{-1} = c^{-1}d \prod_{n=1}^{\lfloor k-1/2 \rfloor} d_{2n} \quad (\prod_{n=1}^{\lfloor k/2 \rfloor} d_{2n-1})^{-1}$

(4) $d^{-1}c^{-1} = c^{-1} \prod_{n=1}^{\lfloor k/2 \rfloor} d_{2n-1} \quad (\prod_{n=1}^{\lfloor k-1/2 \rfloor} d_{2n})^{-1}d$

Remark 2.2.6:

This lemma defines a collecting process which writes any element of a nilpotent group of class k in terms of basic

commutators of weight $\leq k$ that will be ordered from left to right by the weight ordering introduced above. That is we pick out the leftmost, smallest commutator that has not yet been collected and move it to the left using the above four rules. So if G is a free nilpotent group of class k and w is a reduced word in G_{k-1} then w may be effectively written as a product of the elements of the free basis of G_{k-1} given by Lemma 2.2.4.

The following lemma appears in much of the literature but we present a relatively straightforward proof for the sake of completeness.

Lemma 2.2.7 (c.f.[21] or [53]):

Let G be a nilpotent group of class k. If $x_1,\ldots,x_k$ are elements of G and $x_i = a.b$ for some i then

$$[x_1,\ldots,a.b,\ldots,x_k] = [x_1,\ldots,a,\ldots,x_k].[x_1,\ldots,b,\ldots,x_k].$$

Proof:

First let us introduce the notation C_s by induction:

If $s = 1$ then $C_s = x_1$.

If $1 < s \leq k$ then $C_s = [x_1,..,x_s]$.

We will show that the result holds when $1 < i < k-1$.

Upon seeing this proof it is easy to see that the result also holds for $i = 1,k-1,k$. The notation used below is not defined in these instances and for $i = 1$ we have to initially apply Lemma 2.2.2 (ii) instead of (i) but otherwise the arguments are very similar. For a word w lying in G we will use 'w (mod G_i)' to mean 'w.u' for some u where $u \in G_i$. Before we prove this lemma we make and prove two claims.

Claim 1:

If x,y,z,w are elements of G where x,y are of weight 1 then:

$$[w,xy,z] = [w,x,z][w,y,z] \pmod{G_{wt(w) + wt(z) + 1}}$$

Proof:

$[w,xy,z] = [[w,xy],z]$

$= [[w,y][w,x]^{y},z]$ (by Lemma 2.2.2 (i))

$= [[w,y][y,[w,x]^{-1}][w,x],z]$ (by Lemma 2.2.2 (iii)

$= [[w,y][w,x] \pmod{G_{wt(w) + 1}},z]$ (by Lemma 2.2.5)

$= [w,y,z]\ [w,x] \pmod{G_{wt(w) + 1}}\ [[w,x] \pmod{G_{wt(w) + 1}},z]$

(by Lemma 2.2.2 (ii))

$= [w,y,z][[w,x] \pmod{G_{wt(w)+1}},z] \pmod{G_{2wt(w)+wt(z)+1}}$

(by Lemmas 2.2.2 (ii),2.2.2(iii) and 2.2.5: Note that some of the commutators generated by the use of these lemmas are of weight much greater than 2wt(w) + wt(z) + 2 but this is the smallest of the weights of those commutators)

$= [w,y,z][w,x,z]^{r}[r,z] \pmod{G_{2wt(w) + wt(z) + 1}}$

(by Lemma 2.2.2 (ii) where $wt(r) \geq wt(w) + 2$)

$= [w,y,z][w,x,z] \pmod{G_{wt(w) + wt(z) + 1}}$

(by Lemmas 2.2.2 (iii), 2.2.5 and the facts that $wt([r,[w,x,z]^{-1}]) \geq 2wt(w) + wt(z) +3$ and $wt([r,z]) \geq wt(w) +wt(z) + 2$)

Hence Claim 1 is proven.

Claim 2:

If $1 < i < k-1$ then for $1 \leq s \leq k-2-i$:

$$[[C_{i-1}, a, x_{i+1}, .., x_{i+s}].$$

$$[C_{i-1}, b, x_{i+1}, .., x_{i+s}] \bmod G_{i+s}, x_{i+s+1}, .., x_k]$$

$$= [[C_{i-1}, a, x_{i+1}, .., x_{i+s}, x_{i+s+1}].$$

$$[C_{i-1}, b, x_{i+1}, .., x_{i+s}, x_{i+s+1}] \bmod G_{i+s+1}, x_{i+s+2}, .., x_k]$$

Proof:

$$[[C_{i-1}, a, x_{i+1}, .., x_{i+s}]$$

$$.[C_{i-1}, b, x_{i+1}, .., x_{i+s}] \bmod G_{i+s}, x_{i+s+1}, .., x_k]$$

$$= [[C_{i-1}, a, x_{i+1}, .., x_{i+s}, x_{i+s+1}] [C_{i-1}, b, x_{i+1}, .., x_{i+s}] \bmod G_{i+s}$$

$$.[[C_{i-1}, b, x_{i+1}, .., x_{i+s}] \bmod G_{i+s}, x_{i+s+1}], x_{i+s+2}, .. x_k]$$

(by Lemma 2.2.2 (i))

$$= [[C_{i-1}, a, x_{i+1}, .., x_{i+s}, x_{i+s+1}]$$

$$.[C_{i-1}, b, x_{i+1}, .., x_{i+s}] \bmod G_{i+s}, x_{i+s+1}] \bmod G_{i+s+1}, x_{i+s+2}, .. x_k]$$

(by Lemmas 2.2.2 (iii) and 2.2.5. As in the proof of Claim 1 most of the commutators generated here are of weight greater than is required but they are in any case of weight not less than $i + s + 2$)

$$= [[C_{i-1}, a, x_{i+1}, .., x_{i+s}, x_{i+s+1}]$$

$$.[C_{i-1}, b, x_{i+1}, .., x_{i+s}, x_{i+s+1}] \bmod G_{i+s+1}, .. x_k]$$

(by Lemmas 2.2.2 (i), 2.2.2 (iii) and 2.2.5. Once again we generate commutators of weight not less than $i + s + 2$)

Hence Claim 2 is proven.

Now if $1 < i < k-1$ then we have that $[x_1,..,x_k]$

$= [[x_1,..,x_{i-1}],ab,x_{i+1},..,x_k]$

$= [[C_{i-1},ab,x_{i+1}],x_{i+2},..,x_k]$

$= [[[C_{i-1},a],x_{i+1}][[C_{i-1},b],x_{i+1}] \bmod G_{i+1},..,x_k]$

(by Claim 1)

$= [[x_1,..,a,..,x_{k-1}][x_1,..,b,..,x_{k-1}] \bmod G_{k-1},x_k]$

(by Claim 2)

$= [x_1,..,a,..,x_k][x_1,..,b,..,x_{k-1}] \bmod G_{k-1}$

$.[[x_1,..,b,..,x_{k-1}] \bmod G_{k-1},x_k]$

(by Lemma 2.2.2 (i))

$= [x_1,..,a,..,x_k][[x_1,..,b,..,x_{k-1}] \bmod G_{k-1},x_k] \bmod G_k$

(by Lemmas 2.2.2(iii) and 2.2.5 generating commutators of weight not less than k +1)

$= [x_1,..,a,..,x_k][[x_1,..,b,..,x_k] \bmod G_k$

(by Lemmas 2.2.2(i),2.2.2(iii) and 2.2.5 again generating commutators of weight not less than k + 1)

$= [x_1,..,a,..,x_k][[x_1,..,b,..,x_k]$ since we are working in a nilpotent group of class k.

When considering unification for nilpotent groups the terms arising will be allowed to contain both variables and constants. For this reason the natural "free object" to consider is as in the next lemma. In this Lemma (and throughout the rest of the thesis) we will take **Z** to be the set of integers and **N** to be the set of non-negative integers.

Lemma 2.2.8:

Let G be the free nilpotent group of class k on constants $\{a_n: n \in N\}$ and variables $\{x_n: n \in N\}$. Let $w \in G_{k-1}$ and $h > 1$. There exists an algorithm to test whether or not there is a substitution for the variables of w under which it becomes an hth power of a member of G_{k-1}, and which determines all such substitutions if they exist.

Proof:

By Lemma 2.2.4, G_{k-1} is free abelian. Let B_k be the basis of G_{k-1}. By lemma 2.2.4 B_k exists and is the set of all basic commutators of weight k on the generators $\{a_n: n \in N\} \cup \{x_n: n \in N\}$.

Now w may be effectively written as a word in this basis by the collecting process as described in Lemma 2.2.5. and Remark 2.2.6.

Thus $w = b_0{}^{r_0}....b_m{}^{r_m}$ where $b_i \in B_k$ and $r_i \in Z$. Now the only members of $\{a_n: n \in N\} \cup \{x_n: n \in N\}$ occurring in the b_i are those occurring in w. So without loss of generality let $a_0,...a_k$ and $x_0,...x_n$ be the members of $\{a_n: n \in N\}$ and $\{x_n: n \in N\}$ respectively that occur in w.

Since $w \in G_k$ the required substitution will be of the form $\theta = \{a_0{}^{i(0,j)}....a_k{}^{i(k,j)}z \bmod G_1 / x_j : j = 0,...,n\}$ where z is a variable standing for words in the generators not appearing in w. We now use Lemma 2.2.7 to write $w\theta$ as a product of basic commutators. The powers of the commutators are polynomials in

$i_{(0,j)},\ldots,i_{(k,j)}$. All substitutions are obtained by inspection of these polynomials (and the commutators involving z). It is enough to determine all solutions mod h, and there are only finitely many of these to test.

2.2.9 Examples:

2.2.9.1: Let w be the word [a,x][a,b] (where a,b are constants and x is a variable) lying in G the free nilpotent group of class k introduced above where k = 2.

Suppose we wish to express w as a cube. We make the substitution a^ib^jz mod G_1/x. This gives us $[a,b]^{j+1}[a,z]$.

Therefore i is arbitrary, j = 3k-1 and z is a cube.
So all substitutions are given in the format:

$a^ib^{3k-1}z^3$ mod G_1/x

2.2.9.2: Let w be the word [a,x][a,b][a,y] (where a,b are constants and x,y are variables) lying in G the free nilpotent group of class k introduced above where k = 2. Suppose we wish to express w as a square. We make the substitution $\{a^ib^jzu/x$, $a^nb^msv/x\}$ where $u,v \in G'$.
This gives us $[a,b]^{j+m+1}[a,zs]$.

Therefore i is arbitrary, n is arbitrary, m is arbitrary, j = 2k - m -1 and zs is a square i.e. $z = t^2s^{-1}$.
So all substitutions are given in the format:
$\{a^ib^{2k-m-1}t^2s^{-1}u/x$, $a^nb^msv/x\}$ where $u,v \in G'$.

2.3 Solution of equations in free nilpotent groups of class k

2.3.1 Remark:

The algorithm that we derive is relative to having an algorithm which generates the complete, minimal solution set of any system of nth power free diophantine equations of degree n for $1 \leq n \leq k$. We will call such an algorithm A_k.

2.3.2 A description of the algorithm to find the complete, minimal solution set of any equation in a free nilpotent group of class k:

In this section we show, relative to having some A_k, how to solve arbitrary equations in G, a free nilpotent group of class k. Now any equation $w_1 = w_2$ in a group can be recast as $w_1 w_2^{-1} = 1$, so we let w be any word in G, also including unknowns, and consider how we may solve $w = 1$. If we take $\{a_i : a_i < N\}$ as a free basis for G (where N is the rank of G, either finite or $\aleph_0$).

Then w will be a word in the a_i, and finitely many variables $x_0, \ldots, x_{n-1}$ so may be viewed as a word in the free nilpotent group H of class k on $\{a_i : a_i < N\} \cup \{x_i : i \in N\}$ as basis.

To solve the given problem in the free nilpotent group of class k, and show that (where the word to be unified with 1 does not lie in G_1) each solution is of the form

$$t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-1)} / x_0, \ldots, t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-1)} / x_{n-1}$$

where t_i / x_i is the general solution of the homogenised

abelianized version of w and $v_{(i,j)}/x_i$, $1 \leq i \leq n-1$, is a particular solution of $w = 1$ in the quotient group G/G_j, we use induction on k.

(Note that the t_i are words in constants and variables. We can assume that the variables of the t_i are new ones not already occurring in w).

For $k = 1$ the group is abelian and we can apply Theorem 1.5.7. This algorithm is relative to having an algorithm which finds all the solutions to linear diophantine equations. See [28],[35] for the presentation of such an algorithm.

Considering the form of the solution we note that w will be of the form:

$$x_0^{q_0} \ldots x_{n-1}^{q_{n-1}} a_0^{r_0} \ldots a_{m-1}^{r_{m-1}}$$

where q_i, r_i is the exponent sum of x_i, a_i respectively in w.

The most general solution as provided by Theorem 1.5.7 is of the form:

$$t_0 v_{(0,0)}/x_0, \ldots, t_{n-1} v_{(n-1,0)}/x_{n-1}$$

where, in the theory of abelian groups, t_i/x_i is the most general solution to the homogeneous equation $x_0^{q_0} \ldots x_{n-1}^{q_{n-1}} = 1$ and $v_{(i,0)}/x_i$ is a particular solution to $w = 1$.

For the induction step we first consider the problem $w = 1$? in G/G_{k-1} and solve (if possible) using the algorithm given by the induction hypothesis, since G/G_{k-1} is a nilpotent group of

class k - 1. (If that tells us there is no solution, then there cannot be a solution in G either). Suppose the algorithm for k-1 gives a complete, minimal solution set M_{k-1}. We consider each member of M_{k-1} in turn.

Suppose $\theta \in M_{k-1}$. Then we assume inductively that θ is of the form

$$t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)} / x_0, \ldots, t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)} / x_{n-1}$$

where t_i/x_i is the general solution of the abelianized version of w and $v_{(i,j)}/x_i$, $1 \leq i \leq n-1$, is a particular solution of $w = 1$ in the quotient group G/G_j as given by the induction hypothesis.

Since we have solved the equation in G/G_{k-1}, for any solution to the main problem we must have

$$x_0 \in t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)} H_{k-1},$$

$$x_1 \in t_1 v_{(1,0)} v_{(1,1)} \cdots v_{(1,k-2)} H_{k-1},$$

$$\vdots$$

$$x_{n-1} \in t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)} H_{k-1}.$$

$$\text{i.e. } x_0 = t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)} u_0$$

$$x_1 = t_1 v_{(1,0)} v_{(1,1)} \cdots v_{(1,k-2)} u_1$$

$$\vdots$$

$$x_{n-1} = t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)} u_{n-1}$$

where $u_i \in H_{k-1}$, $0 \leq i \leq n-1$.

Therefore we must have

$$w(t_0u_0v_{(0,2)}\cdots v_{(0,k-2)}u_0, \ldots, t_{n-1}u_{n-1}v_{(n-1,2)}\cdots v_{(n-1,k-2)}u_{n-1}$$

$$, a_0, \ldots, a_{m-1}) = 1 \text{ in } H. \text{ But } u_i \in H_{k-1} \text{ which lies in the centre of}$$

H. Therefore

$$u_0^{q_0} \cdots u_{n-1}^{q_{n-1}} =$$

$$w^{-1}(t_0u_0v_{(0,2)}\cdot\cdot v_{(o,k-2)}, \cdot\cdot, t_{n-1}u_{n-1}v_{(n-1,2)}\cdot\cdot v_{(n-1,k-2)}, a_0, \ldots, a_{m-1})$$

The normal way to solve this (as in Theorem 1.5.7) is first, to determine a "general" solution to the homogeneous equation

$$u_0^{q_0} \cdots u_{n-1}^{q_{n-1}} = 1,$$

and second, to determine a "particular" solution of the given equation. The two solutions are then multiplied together to find the general solution to the original equation. In this instance the general solution to the homogeneous equation is, however, subsumed in the solution already found as we shall see below (Theorem 2.3.3) and it suffices to find a particular solution.

We have $u_0^{q_0} \cdots u_{n-1}^{q_{n-1}} = w^{-1}$.

Now we calculate $h = hcf(q_0, \ldots, q_{n-1})$.

If h=1 we have a solution determined as follows:

We know that $1 = c_0q_0 + c_1q_1 + \ldots + c_{n-1}q_{n-1}$ for appropriate integers c_i by Euclid's algorithm. Then

$$u_0 = w^{-c_0}, \; u_1 = w^{-c_1}, \; \ldots, u_{n-1} = w^{-c_{n-1}}$$

provides us with a particular solution.

If h>1 we may or may not have a solution depending upon whether w is an hth power or *can be made into one* by substituting for the

variables of the t_i's.

We can use Lemma 2.2.5 to see if w *is* an hth power. If so then $w^{-1} = s^h$ for some word s. Once again we find integers c_i such that $h = c_0q_0 + c_1q_1 + \ldots + c_{n-1}q_{n-1}$ and this time we find that $u_0 = s^{c_0}$, $u_1 = s^{c_1}, \ldots, u_{n-1} = s^{c_{n-1}}$ provides a particular solution.

If w is not an hth power we can attempt to make it so by the algorithm of Lemma 2.2.8 and then apply each resulting substitution σ to obtain $v_{(i,k-1)}\sigma/u_i$. Lemma 2.2.8 gives all substitutions which can express w as an hth power. If S is the set of all such substitutions then all the solutions to the problem are given by $x_i = (t_i v_{(i,o)} v_{(i,1)} \cdots v_{(i,k-2)} v_{(i,k-1)})\sigma$ for every $\sigma \in S$.

If we cannot make w an hth power then there can be no solution. Once we have a particular solution

$$v_{(0,k-1)}/u_0, \ldots, v_{(n-1,k-1)}/u_{n-1}$$

then the general solution to the main problem is given by

$$x_0 = t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)} v_{(0,k-1)}$$

$$\vdots$$

$$x_{n-1} = t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)} v_{(n-1,k-1)}$$

remembering that we still have to show that the "general" part of the solution in G/G_{k-1} is subsumed in the "general" part of the solution of the abelianized version (see Theorem 3.2.2). The form of this solution is as required to fit in with our inductive assumption.

However, we have so far overlooked the case when $u_0^{q_0} \ldots u_{n-1}^{q_{n-1}}$ is trivially equal to the identity i.e. h = 0.

Remark 2.3.2.1:

This occurs when the exponent sums of the u_i are zero.

We readily deduce that h = 0 if and only if $w \in H_1$.

Now *if h=0* then we have to solve in G the problem

$$w(t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)}, \ldots, t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)},$$

$$a_0, \ldots, a_{m-1}) = 1.$$

We know that $t_i v_{(i,0)} v_{(i,1)} \cdots v_{(i,k-2)}$ is a word in variables and constants. Suppose the variables occurring in

$$w(t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)}, \ldots, t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)},$$

$$a_0, \ldots, a_{m-1})$$

are $\{z_0, \ldots, z_{p-1}\}$. This means that we can assume that the equation to be solved is of the form

$$w'(z_0, \ldots, z_{p-1}, a_0, \ldots, a_{m-1}) = 1.$$

Let B(k) be the set of basic commutators of weight $\leq k$ on $\{a_0, \ldots, a_{m-1}\}$.

By Lemma 2.2.4 we have that any solution will be of the form $z_i = b_0^{z(i,0)} \ldots b_{g-1}^{z(i,g-1)} s_i$ where z(i,j) is an integer variable, $b_i \in B(k)$, $g = |B(k)|$ and s_i is a variable standing for all the basic commutators not involving $a_0, \ldots, a_{m-1}$.

If we substitute this into $w'(z_0, \ldots, z_{p-1}, a_0, \ldots, a_{m-1}) = 1$, use Lemma 2.2.7 and write as a product of the basis of G_{k-1} then the

powers of the basic commutators will be kth power free diophantine polynomials of degree k (there will be no kth powers since to obtain one there would have to be a commutator of weight k of the form $[a_j^{z(i,j)},\ldots,a_j^{z(i,j)}]$ and this is trivially the identity).

Now equating the basic commutators on the left hand side with 1 we obtain a system of kth power free diophantine equations of degree k. We now apply algorithm A_k and obtain all the solutions to w = 1. The conditions for s_i can be given by inspection of the commutators in which they appear (see example 2.7.8).

We now justify the claim that we only require a particular solution for the u_i in the algorithm.

Theorem 2.3.3:

Suppose we have a minimal algorithm to solve equations in a finitely generated free nilpotent group of class k-1. Suppose also that we have the problem $w(x_0,\ldots,x_{n-1},a_0,\ldots,a_{m-1}) = 1$ in G the nilpotent group of class k defined in algorithm 2.3.2 such that $w(x_0,\ldots,x_{n-1},a_0,\ldots,a_{m-1}) \notin H_1$.

The general solution to $u_1q_1\ldots u_nq_n = 1$ in the above algorithm is subsumed in the general solution to the abelian case.

Proof:

Apply algorithm 2.3.2 and obtain $h = hcf(q_0,\ldots,q_{n-1})$.

Since $w(x_0,\ldots,x_{n-1},a_0,\ldots,a_{m-1}) \notin H_1$ we deduce from Remark 2.3.2.1 that $h \neq 0$.

Hence algorithm 2.3.2 gives us a solution set in G of:

$$S = \{\theta = \{t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-1)}/x_0, \cdots$$

$$\cdots, t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-1)}/x_{n-1}\}\}$$

where $\{t_0 v_{(0,0)} v_{(0,1)} \cdots v_{(0,k-2)}/x_0, \cdots$

$$\cdots, t_{n-1} v_{(n-1,0)} v_{(n-1,1)} \cdots v_{(n-1,k-2)}/x_{n-1}\} \in M_{k-1}$$

and $\{v_{(0,k-1)}/x_0, \ldots, v_{(n-1,k-1)}/x_{n-1}\}$ is the particular solution generated by algorithm 2.3.2.

Suppose we have that $\sigma = \{\sigma_0/x_0, \ldots, \sigma_{n-1}/x_{n-1}\}$ is another solution of $w = 1$. We must show that for some $\theta \in S$ there exists a substitution λ such that $\theta.\lambda = \sigma$.

We construct each

$\theta = \{t_0 v_{(0,0)} \cdots v_{(0,k-1)}/x_0, \cdots, t_{n-1} v_{(n-1,0)} \cdots v_{(n-1,k-1)}/x_{n-1}\}$ as in Algorithm 2.3.2.

As $w(\sigma_0, \ldots, \sigma_{n-1}, a_0, \ldots, a_{m-1}) = 1$ it follows by the induction hypothesis that there is a substitution ρ and

$\{t_0 v_{(0,0)} \cdots v_{(0,k-2)}/x_0, \cdots, t_{n-1} v_{(n-1,0)} \cdots v_{(n-1,k-2)}/x_{n-1}\} \in M_{k-1}$

(corresponding to some $\theta \in S$) such that in G/G_{k-1}

$\{t_0 v_{(0,0)} \cdots v_{(0,k-2)}/x_0, \ldots, t_{n-1} v_{(n-1,0)} \cdots v_{(n-1,k-2)}/x_{n-1}\}\rho$

$= \{\sigma_0/x_0, \ldots, \sigma_{n-1}/x_{n-1}\}$

Let $\sigma_i = r_i\ (t_i v_{(i,0)} \cdots v_{(i,k-2)})\rho$ where $r_i \in G_{k-1}$

$= r_i\ (t_i\rho) \cdot (v_{(i,0)} \cdots v_{(i,k-2)})\rho$

$\therefore r_0 q_0 \cdots r_{n-1} q_{n-1} w((t_0\rho)(v_{(0,0)} \cdots v_{(0,k-2)})\rho, \cdots$

$\cdots, (t_{n-1}\rho)(v_{(n-1,0)} \cdots v_{(n-1,k-2)})\rho, a_0, \cdots a_{m-1}) = 1$

Also, by the construction of the $v_{(i,k-1)}$ in Algorithm 2.3.2

(using the member of M_{k-1} introduced above) we have that

$$v_{(0,k-1)}{}^{q_0} \cdots v_{(n-1,k-1)}{}^{q_{n-1}}$$

$$w(t_0 v_{(0,0)} \cdots v_{(0,k-2)}, \ldots, t_{n-1} v_{(n-1,0)} \cdots v_{(n-1,k-2)}, a_0, \ldots a_{m-1}) = 1$$

Applying the substitution ρ we derive

$$((v_{(0,k-1)}\rho)^{-1} r_0)^{q_0} \ldots ((v_{(n-1,k-1)}\rho)^{-1} r_{n-1})^{q_{n-1}} = 1.$$

Now t_i/x_i is the most general solution of the homogeneous equation $x_0^{q_0} \ldots x_{n-1}^{q_{n-1}} = 1$ in the theory of abelian groups.

Therefore there is a substitution τ such that

$$(v_{(i,k-1)}\rho)^{-1} r_i = t_i \tau$$

Now $\sigma_i = r_i (t_i \rho)(v_{(i,0)}\ v_{(i,1)} \cdots v_{(i,k-2)})\rho$

$$= v_{(i,k-1)}\rho \cdot t_i(\tau) t_i(\rho) \cdot (v_{(i,0)}\ v_{(i,1)} \cdots v_{(i,k-2)})\rho$$

$$= v_{(i,k-1)}\rho \cdot (t_i \lambda) \cdot (v_{(i,0)} v_{(i,1)} \cdots v_{(i,k-2)})\rho$$

(where $y\lambda = y(\tau)y(\rho)$ for each (new) variable y of the t_i.

Notice that τ substitutes only members of G_{k-1} for y)

$$= (t_i v_{(i,0)} v_{(i,1)} \cdots v_{(i,k-1)})\lambda$$

$\therefore$ we have constructed λ such that $\theta\lambda = \sigma$.

Algorithm 2.3.2 is an algorithm to solve equations in any finitely or infinitely generated free nilpotent group of class k. However, if we are considering unification in the theory $N_k G$ then it is the case of infinitely generated free nilpotent groups of class k which is relevant. We will now show that $N_2 G$ is nullary and indicate why it is very likeley that $N_k G$ is nullary.

Proposition 2.3.4:

Suppose H is a free nilpotent group of class k generated by the infinite set $\{a_0, a_1, a_2, \ldots\} \cup \{x_0, x_1, x_2, \ldots\}$. If $w_1, \ldots, w_n$ are finitely many words in H_{k-1} then there is $b \in H_{k-1}$ which is not an instance of any w_i.

Remark 2.3.5:

The following lemma is a proof of this proposition for case $k = 2$. We use it to show that N_2G is nullary.

It seems very likely that the proposition is true for all k. If so then it is a straightforward task to show that N_kG is nullary.

Lemma 2.3.6:

Suppose H is a free nilpotent group of class 2 generated by the infinite set $\{a_0, a_1, a_2, \ldots\} \cup \{x_0, x_1, x_2, \ldots\}$. If $w_1, \ldots, w_n$ are finitely many words in H' then there is $b \in H'$ which is not an instance of any w_i.

Proof:

Let k be such that each w_i is a product of $\leq k$ commutators. Consider $b = [a_0, a_1][a_2, a_3]\ldots[a_{2k}, a_{2k+1}] \in H'$ and suppose by way of contradiction that there exists a substitution σ such that $b = w\sigma$ where $w = w_i$ some i. Let $w = [u_1, v_1][u_2, v_2]\ldots[u_k, v_k]$. As before, we may suppose that the words substituted for each x_i contain only constants a_j for $j \leq 2k+1$ and that no other

constants appear in w. Let $l(i,j)$, $m(i,j)$ be the exponent sums of a_i in u_j, v_j respectively. Expanding using Lemma 2.2.7 and collecting like terms (using H' abelian) we find that

$$[u_j, v_j]\sigma = \Pi_{0\le i,i'\le 2k+1}[a_i, a_{i'}]^{l(i,j)m(i',j)}.$$

Equating this with b and comparing coefficients we find that for $i \le i'$

$$\Sigma_{1\le j\le k}(l(i,j)m(i',j) - l(i',j)m(i,j)) = \begin{cases} 1 & \text{if } i \text{ is even and } i' = i+1 \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathbf{l}_j = \begin{bmatrix} l(0,j) \\ \vdots \\ l(2k+1,j) \end{bmatrix}$ and $\mathbf{m}_j = \begin{bmatrix} m(0,j) \\ \vdots \\ m(2k+1,j) \end{bmatrix}$

We therefore obtain

$$\mathbf{l}_1\mathbf{m}_1^T - \mathbf{m}_1\mathbf{l}_1^T + \ldots + \mathbf{l}_k\mathbf{m}_k^T - \mathbf{m}_k^T\mathbf{l}_k =$$

$$\begin{bmatrix}
0 & 1 & 0 & . & . & . & . & . & . & . & 0 & 0 \\
-1 & 0 & 0 & . & . & . & . & . & . & . & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & . & . & . & . & . & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & . & . & . & . & . & 0 & 0 \\
. & . & . & 0 & 0 & 1 & 0 & . & . & . & 0 & 0 \\
. & . & . & 0 & -1 & 0 & 0 & . & . & . & 0 & 0 \\
. & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . \\
0 & 0 & . & . & . & . & . & . & 0 & 1 & 0 & 0 \\
0 & 0 & . & . & . & . & . & . & -1 & 0 & 0 & 0 \\
0 & 0 & . & . & . & . & . & . & 0 & 0 & 0 & 1 \\
0 & 0 & . & . & . & . & . & . & 0 & 0 & -1 & 0
\end{bmatrix}$$

a $2(k+1) \times 2(k+1)$ non-singular matrix.

Now $\text{rank}(l_j m_j^T)$, $\text{rank}(m_j l_j^T) \leq 1$ for each j, so

$\text{rank}(l_1 m_1^T - m_1 l_1^T + \ldots + l_k m_k^T - m_k l_k^T) \leq 2k$, a contradiction.

Theorem 2.3.7:

N_2G is nullary.

Proof:

By Theorem 1.4.2 we can restrict our attention to free nilpotent groups of class 2.

Consider the word $[x, a_0]$ in G, the free nilpotent group of class 2 generated by the infinite set $\{a_0, a_1, \ldots\}$.

Now any solution to $w = 1$ may be written in the form $x = \prod_{i<m} a_i^{r_i} c$

where $c \in G'$.

Substituting this into the given problem we deduce that $r_i = 0$ for $i > 0$ so that the most general solution to $w = 1$ is

$$x = a_0^r c, \quad r \in Z, \quad c \in G'.$$

Suppose that t/x is any unifier of the problem. Then $t = a_0^r c$ where c is some product of m commutators. But t is an instance of another solution $t' = a_0^s c'$ with c' a product of m+1 commutators. However, by Lemma 2.3.6 t' cannot be an instance of t since c' is not an instance of c.

Hence $[x, a_0] = 1$ has no minimal solution in G.

2.3.8 Remark:

First order nullary theories are few and far between. See 1.3 for a brief survey of the ones that are known. In the above theorem we showed that N_2G can be added to the list of first

order nullary theories. We also indicated why it is very likeley that each member of the infinite class of theories $\{N_kG: 1 < k \ \& \ k \in \mathbf{N}\}$ is nullary.

We will now take a closer, more detailed look at nilpotent groups of class 2:

2.4 Unification in nilpotent groups of class 2

It follows from Definition 2.2.3 that a group G is nilpotent of class 2 if there exists a normal subgroup N of G such that G/N is abelian and the members of N commute with all members of G (N is *central*) and that N can be taken to be the commutator subgroup of G, written G'. It also follows that N_2G, the theory of nilpotent groups of class 2, is the group axioms together with [[x,y],z] = 1.

Now by algorithm 2.3.2 we have an algorithm for finitely generated free nilpotent groups of class 2 provided we have an algorithm to generate all the solutions of systems of square free diophantine equations of degree 2. However, the equations that actually occur in the algorithm take a very specialised form. We will call a square free diophantine equation of degree 2 that has this form a τ-equation. We will look at these more closely later on. In Chapter 3 we show (when showing that the unification problem is undecidable) that the solution of equations in a finitely generated free nilpotent group of class 5 is recursively insoluble. This rests upon the fact that for $k \geq 5$ the problem of algorithmically finding integer solutions to the equations that occur is insoluble by the result of Matiyasevitch [50]. Clearly the higher the value of k we take (for $2 \leq k \leq 4$) the less likely we are of constructing a successful complete, minimal algorithm for finitely generated free nilpotent groups of class k. This is one of the reasons that we concentrate on $k = 2$. We have already shown in Theorem 2.3.10 that a first order theory of nilpotent groups of class 2 is nullary. However, in the following section we show that by working in a two-sorted theory the cardinality type of the theory becomes either infinitary or the cardinality type of any system of τ-equations. Finally we take a brief look

at problems in only one variable in a nilpotent group of class 2 as looked at by Repin [57]. But firstly we define τ-equations:

2.4.1 Definition:

Suppose we have a set of integer variables $\{z(i,j): 1 \leq i \leq n, 1 \leq j \leq m\}$ and an integer constant K then a *τ-polynomial* is a diophantine polynomial of the form:

$$K \pm \sum_{\substack{1\leq i\leq n \\ 1\leq j\leq m}} z(i,j) + \sum_{\substack{i\neq k \\ j<l}} z(i,j)z(k,l) - z(i,l)z(k,j).$$

If p is a τ-polynomial then p = 0 is a *τ-equation*.

Theorem 2.4.2:

The equations that occur in algorithm 2.3.2 for k = 2 are τ-equations.

Proof:

Suppose G is a free nilpotent group of class 2 on $\{a_0, a_1, \ldots\}$. Suppose also that we have to solve the problem $w(x_0, \ldots, x_{n-1}, a_0, \ldots, a_{m-1}) = 1$ in G. As before, w is a word in the a_i, and finitely many variables $x_0, \ldots, x_{n-1}$ so may be viewed as a word in the free nilpotent group H of class k on $\{a_i : a_i < N\} \cup \{x_i : i \in N\}$ as basis.

Algorithm 2.3.2 will generate solutions to the abelianized version of w of the form

$$t_0u_0/x_0, \ldots, t_{n-1}u_{n-1}/x_{n-1}$$

where t_i/x_i, u_i/x_i are the general and particular solutions respectively of the abelianized version of w.

The problem is first solved in the quotient group G/G' and the problem is lifted to G by noting that any solution to the main problem must be of the form $x_i = t_i u_i v_i$, $0 \leq i \leq n-1$, where $v_i \in H'$.

We apply algorithm 2.3.2 and the square free diophantine equations of degree 2 that occur do so only when $v_0^{q_0} \ldots v_{n-1}^{q_{n-1}}$ is trivially equal to the identity

i.e. $h = 0$.

When $h = 0$ we have to solve in G the problem

$$w(t_0u_0, \ldots, t_{n-1}u_{n-1}, a_0, \ldots, a_{m-1}) = 1.$$

Now t_i is a word in variables and constants. Suppose the variables occurring in $w(t_0, \ldots, t_{n-1}, a_0, \ldots, a_{m-1})$ are $\{z_0, \ldots, z_{k-1}\}$.

Hence we have to solve $w'(z_0, \ldots, z_{k-1}, a_0, \ldots, a_{m-1}) = 1$. However $w'(z_0, \ldots, z_{k-1}, a_0, \ldots, a_{m-1})$ can be seen as a word in T the free nilpotent group of class 2 on generators

$$\{a_0, \ldots, a_{m-1}\} \cup \{z_0, \ldots, z_{k-1}\}.$$

By Lemma 2.2.4, T' is free abelian with basis

$$\{[a_i, a_j] : i > j\} \cup \{[a_i, z_j] : i, j \in N\} \cup \{[z_i, z_j] : i > j\}.$$

Now since $w'(z_0, \ldots, z_{k-1}, a_0, \ldots, a_{m-1}) \in T'$ it may be written as a word in this basis. Thus for some N and integers $l(i,j)$, $m(i,j)$, $n(i,j)$ we have

$$\Pi\{[a_i, a_j]^{l(i,j)} : 0 \leq j < i < N\}.$$

$$\Pi\{[a_i, z_j]^{m(i,j)} : 0 \leq i, j < N\} . \Pi\{[z_i, z_j]^{n(i,j)} : 0 \leq j < i < N\} = 1$$

By Lemma 2.2.7 and T' abelian we have that any solution will be

of the form $z_i = a_0^{z(i,0)} \ldots a_{m-1}^{z(i,m-1)} s_i \bmod G'$ where $z(i,j)$ is an integer variable and s_i is a variable standing for the a_j not occurring in w.

If we substitute this into

$$\Pi\{[a_i,a_j]^{l(i,j)}: 0 \le j < i < N\}.\Pi\{[a_i,z_j]^{m(i,j)}:0 \le i,j < N\},$$

use Lemma 2.2.7 and write as a product of the basis of G' then the powers of the basic commutators will be linear diophantine polynomials.

However, we still have to consider $\Pi\{[z_i,z_j]^{n(i,j)}:0 \le j < i < N\}$.

Substituting $z_i = a_0^{z(i,0)} \ldots a_{m-1}^{z(i,m-1)} s_i \bmod G'$ yields

$$\Pi\{[a_0^{z(i,0)} \ldots a_{m-1}^{z(i,m-1)} s_i,$$

$$a_0^{z(j,1)} \ldots a_{m-1}^{z(j,m-1)} s_i]^{n(i,j)}:0 \le j < i < N\}$$

Now $[a_0^{z(i,0)} \ldots a_{m-1}^{z(i,m-1)}, a_0^{z(j,1)} \ldots a_{m-1}^{z(j,m-1)}]^{n(i,j)}$ can be written in the form $\Pi\{[a_p,a_q]^{\alpha(p,q)}: 0 \le q < p < N\}$ where

$$\alpha(p,q) = n(i,j)(z(i,p)z(j,q) - z(j,p)z(i,q)).$$

Hence substituting $z_i = a_0^{z(i,0)} \ldots a_{m-1}^{z(i,m-1)} s_i \bmod G'$ into

$$\Pi\{[a_i,a_j]^{l(i,j)}: 0 \le j < i < N\}.$$

$$\Pi\{[a_i,z_j]^{m(i,j)}:0 \le i,j < N\}.\Pi\{[z_i,z_j]^{n(i,j)}:0 \le j < i < N\} = 1$$

and equating the basic commutators on the left hand side with 1 we obtain a system of τ-equations as the square free diophantine equations of degree 2 that occur in algorithm 2.3.2.

Now by this theorem we know that if we have an algorithm to generate all solutions of systems of τ-equations then we have an algorithm to generate all the solutions of any equation in G (a free nilpotent group of class 2 on $\{a_0,a_1,\ldots\}$). However, by

Theorem 2.3.10 we cannot generate a most general solution. We now consider a method of getting round this problem by passing to a two sorted theory where the sorts pick out the elements of the derived subgroup. This results in the cardinality type being entirely dependent upon the τ-systems rather than being nullary.

2.4.4 Remark:

Suppose we consider the two-sorted theory MN_2G constructed by adding to N_2G a new set of variables $V'=\{x_1', x_2', \ldots\}$ and the following axioms:

$$\forall x_1 \forall x_2 \exists x_1' (x_1^{-1} x_2^{-1} x_1 x_2 = x_1')$$

$$\forall x_1 \forall x_1' (x_1' x_1 = x_1 x_1')$$

$$\forall x_1' \forall x_2' \exists x_3' (x_1' x_2'^{-1} = x_3').$$

Let G be a free model of MN_2G and suppose that we wish to solve the problem w = 1 in G. The added axioms ensure that the variables of V' stand for the elements of G'. Thus, if we have an algorithm to find all the solutions of τ-systems then we have an algorithm to find all the solutions to problems in MN_2G.

2.4.5 Theorem:

If the cardinality type of The cardinality type of any system of τ-equations is nullary then MN_2G is nullary.

If the cardinality type of any system of τ-equations is $<$ nullary then MN_2G is infinitary.

Proof:

Suppose we have the problem w = 1 lying in G a free model of MN_2G where $w \in H'$ (remember H is the free nilpotent group that has as generators the generators of G together with the variables

of w). We have seen above how to reduce this problem to a system of τ-equations. Also we have seen that for any such solution the corresponding most general solution (for each variable x of w) is of the form $x = a_0^{r_0} \ldots a_{m-1}^{r_{m-1}} c$ where $a_0, \ldots, a_{m-1}$ are the generators of G appearing in w, $r_0, \ldots, r_{m-1}$ are integers and c is an arbitrary product of commutators. We saw in Theorem 2.3.10 that the general form of c could not be represented in the single sorted case. However, here we have a sort available to express this general form. So the cardinality type of MN_2G is dependent upon the cardinality type of τ-systems. However, in example 2.7.4 we present a problem which in MN_2G generates an infinite set of most general unifiers. So the cardinality type of MN_2G is at least infinitary.

Now let us briefly consider solving equations in just one variable in a finitely generated free nilpotent group of class 2 as considered by Repin [57]. Repin does not present his work in terms of unification theory. He works solely in the free object (which ammounts to the same thing). Also he does not look at the cardinality type of the particular class of problems under consideration.

We will present a unification algorithm for nilpotent groups of class 2 with respect to problems in just one variable. Although this has, in effect, already been done by Repin we will present the algorithm in terms of the work presented earlier in this chapter. Also considered will be the cardinality type of this restricted class of problems in the theories of nilpotent groups of class 2 that we have introduced above.

2.4.6 Remark:

Suppose we have the problem $w(a_0,\ldots,a_{m-1},x) = 1$ in the free nilpotent group of class 2, G generated by $\{a_0,\ldots,a_{m-1}\}$ where $w \in G'$. We can view w as a word in the free nilpotent group of class k on $\{a_0,\ldots,a_{m-1}\} \cup \{x\}$. Thus we can (using Lemma 2.2.4) write w as a product of the basic commutators generated by $\{a_0,\ldots,a_{m-1}\} \cup \{x\}$. The commutator $[x,x]$ is trivially the identity so the only basic commutator occuring which involves the variable is of the form $[a_i,x]$. Therefore, substituting

$$x = a_0^{x(i,0)} \ldots a_{m-1}^{x(i,m-1)} \bmod G'$$

where $x(i,j)$ are integer variables and writing as a product of the basic commutators of G we obtain only linear diophantine equations as the powers of the basic commutators. Thus the τ-equations never occur for problems in only one variable. This gives rise to the following theorem.

Theorem 2.4.7:

For problems in only one variable:

(1) The theory N_2G is nullary.

(2) The theory MN_2G is infinitary.

Proof:

(1) The example used in Theorem 2.3.10 to show that N_2G (and hence N_kG) is nullary only used one variable. So restricting to one variable, as far as this is concerned, does not achieve anything.

(1) This follows from Example 2.7.4, Remark 2.4.6. and the fact that there is always a single most general solution to systems of linear diophantine equations.

Remark 2.4.8:

Now in a free nilpotent group of class 2, when considering one variable problems, we have seen that the only basic commutators appearing, that involve the variable, are of the form $[a_i,x]$ so we only obtain linear diophantine equations as the powers of the basic commutators.

It is easy to see that to solve equations involving only one variable in a finitely generated free nilpotent group of class k it is necessary to have some A_{k-1}. This can be seen by considering one variable problems in a nilpotent group of class 3. We can construct a problem where the basic commutator $[a_1,x,x]$ occurs and this gives rise to quadratic diophantine equations as powers of the basic commutators.

We have shown in the last section that N_2G is nullary and that it is likeley that N_kG is nullary. We have also shown that in N_2G we can sidestep this problem by considering MN_2G thus rendering the cardinality type of MN_2G dependent upon the cardinality type of systems of τ-equations and in any case it is at least infinitary. We now show that we can similarly get round the problem (if indeed Proposition 2.3.4 is true) for N_kG by considering a k-sorted theory.

2.5 Unification in k-sorted theories of nilpotent groups of class k

2.5.1 Definition:

In a nilpotent group G of class 3, G_1 is taken to be [G,G] and G_2 is taken to be $[G_1,G]$. N_3G is the group axioms together with [[w,x,y],z] = 1. Let MN_3G be the many-sorted theory constructed by adding to N_3G two new sets of variables $V'=\{x_1',x_2',\ldots\}$, $V''=\{x_1'',x_2'',\ldots\}$ and the following axioms:

$$\forall x_1 \forall x_2 \exists x_1' (x_1^{-1}x_2^{-1}x_1x_2=x_1')$$

$$\forall x_1' \forall x_2 \exists x_1'' (x_1'^{-1}x_2^{-1}x_1'x_2=x_1'')$$

$$\forall x_1 \forall x_1'' (x_1''x_1=x_1x_1'')$$

$$\forall x_1' \forall x_2' \exists x_3' (x_1'x_2'^{-1}=x_3')$$

$$\forall x_1'' \forall x_2'' \exists x_3'' (x_1''x_2''^{-1}=x_3'').$$

In an obvious manner we can define a k-sorted theory of nilpotent groups of class k, MN_kG.

2.5.2 Theorem:

If we have an algorithm to find all solutions of systems of kth power free diophantine equations of degree k then we have an algorithm to find all the solutions to problems in MN_kG. Moreover, the cardinality type of MN_kG is at least infinitary.

Proof:

The sorts pick out the terms of the lower central series rendering the solution of problems in a free model of MN_kG entirely dependent upon the solution of systems of kth power free

diophantine equations of degree k. The algorithm works by first solving in the corresponding free abelian group and then lifting the solution to the corresponding free nilpotent group of class 2 and so on until the solution has been lifted to the free nilpotent group of class k. The algorithm at the nth stage (i.e. in the free nilpotent group of class n) is dependent upon finding a complete, minimal solution set of a given system of nth power free diophantine equations of degree n. Each element of this set must be considered when lifting the solution to class n+1. Thus the cardinality type of MN_kG is infinitary (due to example 2.7.4) or nullary (if the cardinality type of sytems of kth power free diophantine equations of degree k is nullary).

We now relate this work to the work done by U.Martin [47] on special p-groups. We also extend the results to consider nilpotent groups of class k and exponent p.

2.6 Unification in Special p-groups

2.6.1 Definition:

Let p be any prime number. A *special p-group* is a nilpotent group of class 2 together with the identity $x^p = 1$ (i.e. a nilpotent group of class 2 of exponent p). So the *theory of special p-groups* pN_2G is N_2G together with the axiom $\forall x(x^p = 1)$. We can similarly define pMN_2G, the *2-sorted theory of special p-groups*. We can also consider pN_kG (i.e. N_kG together with the axiom $\forall x(x^p = 1)$) and we can turn this into a k-sorted theory to obtain pMN_kG. We will now consider kth power free diophantine equations of degree k where the variables range over finite subsets of **N** and therefore the cardinality type of systems of these equations is either 1 or ω.

2.6.2 Remark:

U. Martin has recently considered unification in special p-groups [47]. Results about special p-groups can, however, also be derived from the above work. The crucial point about special p-groups is that the identity $x^p = 1$ means that we have a finite number of possible solutions of the equations occurring. An algorithm is obtained simply by substituting in every possible solution and checking it. Hence we can easily prove the following theorems.

Theorem 2.6.3:

pN_kG is nullary.

Proof:

the proof follows by Theorem 2.3.10.

Theorem 2.6.4:

pMN_kG is finitary.

Proof:

We cannot have an infinite number of solutions since the p imposes a bound on the power of any element of the group.

We now close this chapter by considering some examples of algorithm 2.3.2 being applied to actual problems.

2.7 Examples and algorithm outline

2.7.1 Outline of the algorithm:

(1) Given a problem $w(x_1,\ldots x_n,a_1,\ldots,a_m) = 1$ lying in G a nilpotent group of class k we first solve for nilpotent of class k-1 to obtain solutions of the form t_i/x_i, if they exist (if no solutions exist for k-1 then there can be no solutions for k either). We then note that solutions to the main problem must be of the form t_iu_i/x_i where $u_i \in G_{k-1}$.

(2) Substituting this into the main problem we obtain

$u_1^{q_1}\ldots u_n^{q_n} = w(t_1,\ldots t_n,a_1,\ldots,a_m)^{-1}$

Now we calculate $h = hcf(q_1,\ldots,q_n)$.

(3) If $h = 0$ we have $w(t_1,\ldots t_n,a_1,\ldots,a_m) = 1$. For every variable occurring we make the substitution in the form given earlier to obtain a system of kth power free diophantine equations of degree k. We must solve this system to obtain the solutions.

(4) If $h = 1$ we find a particular solution for u_i and this suffices since a general solution is subsumed into the general solution for the abelian case.

(5) If $h > 1$ we check whether or not w is an hth power.
If w is an hth power we can find a solution in a similar manner to (4).
If w is not an hth power then we check whether or not it can be made into one. If so then we find all the possible substitutions

and obtain a solution in a similar manner to (4). If w cannot be made into an hth power then there can be no solution.

2.7.2:

Suppose we have the problem xaby[b,a] = 1 in G, a free nilpotent group of class 2, where a,b are constants and x,y are variables. The first step of the method is to abelianize and solve using Theorem 1.5.7. We obtain $x = ta^{-1}$, $y = t^{-1}b^{-1}$.

Therefore any solution in G will be of the form

$x = ta^{-1}u$, $y = t^{-1}b^{-1}v$

where $u,v \in G'$.

Substituting this into the problem we obtain $tbt^{-1}b^{-1}[b,a]uv = 1$.

Therefore we have uv = [a,b][b,t].

We let u = [a,b][b,t] and v = 1 and all solutions are given by

$\theta = \{x = ta^{-1}[a,b][b,t],\ y = t^{-1}b^{-1}\}$

Now note that we could have let u = [a,b] and v = [b,t] (or any other combination). A natural question at this point is whether or not this would generate another most general unifier. We showed in Theorem 2.3.3 that a general solution in G' is subsumed into the general solution for the abelian case. We can see this, in this example, by substituting u = [a,b], v = [b,t]. This yields

$\sigma = \{x = ta^{-1}[a,b],\ y = t^{-1}b^{-1}[b,t]\}$.

However, if we let $\lambda = \{t[t,b]/t\}$ then $\theta\lambda = \sigma$. Also if we let $\lambda = \{t[b,t]/t\}$ then $\sigma\lambda = \theta$. i.e. we have $\theta =_{N_2G} \sigma$.

2.7.3:

Now let us consider the same problem but when G is a free nilpotent group of class 3. We first obtain the solution in the

class 2 case as above. Then any solution in G is of the form

$x = ta^{-1}[a,b][b,t]u,\ y = t^{-1}b^{-1}v$

where $u,v \in G_2$.

Substituting this into the problem we obtain

$ta^{-1}[a,b][b,t]abt^{-1}b^{-1}[b,a]uv = 1.$

Therefore we have $u = [a,b]btb^{-1}a^{-1}[t,b][b,a]at^{-1},\ v = 1$

and all solutions are given by

$\theta = \{x = ta^{-1}[a,b][b,t][a,b]btb^{-1}a^{-1}[t,b][b,a]at^{-1},\ y = t^{-1}b^{-1}\}.$

2.7.4:

Suppose we have the problem $x^2[a,x]y^{-2} = 1$ in G, the free nilpotent group of class 2 generated by {a,b}. Abelianizing we obtain $x = t,\ y = t$ where t is a new variable.

Thus any solution in G is of the form $x = tu,\ y = tv$.

Substituting this into the problem we obtain $u^2v^{-2} = [t,a]$.

We now have to consider substitutions of t to make the RHS into a square. Let $t = a^ib^j \bmod G'$. This gives us $u^2v^{-2} = [b,a]^j$. Therefore i is arbitrary and j is even. So $t = a^ib^{2k}\ r$ (where $r \in G'$).

We let $u = [b,a]^k$, $v = 1$ and we have all solutions given by $x = a^ib^{2k}[b,a]^kr,\ a^ib^{2k}r$ (where $r \in G'$). Notice that in a single sorted structure there does not exist a most general unifier (because we can have arbitrarily long products of commutators. However, in a two sorted structure such as the ones we have been considering there is an infinite number of most general unifiers.

2.7.5:

Suppose we have the problem $x^2a^2[y,b] = 1$ in G, a free nilpotent group of class 2, where a,b are constants and x,y are variables. Abelianizing we obtain $x = a^{-1},\ y = t$ where t is a new

variable.

Thus any solution in G is of the form $x = a^{-1}u$, $y = tv$.

Substituting this into the problem we obtain $u^2 = [b,t]$.

We now have to consider substitutions of t to make the RHS into a square. We let $t = zb^j \bmod G'$. A most general solution is given by z being a square and j being arbitrary. Therefore we let $t = z^2b^j \bmod G'$ and $u = [b,z]$, v is an arbitrary member of G'.

Thus all solutions are of the form $x = a^{-1}\,[b,z]$, $y = z^2b^jr$ (where $r \in G'$).

Note that we can obtain more than one most general unifier in the two-sorted structure (by letting $j = 0$ and $j = 1$).

2.7.6:

Suppose we have the problem $x^3z^6b^6[y,b] = 1$ in G, a free nilpotent group of class 2, where a,b are constants and x,y,z are variables. We abelianize and solve to obtain $x = t^2$, $y = s$, $z = t^{-1}b^{-1}$ where s,t are new variables.

Therefore any solution in G will be of the form

$x = t^2u$, $y = sr$ $z = t^{-1}b^{-1}v$

where $u,r,v \in G'$.

Substituting this into the problem we obtain

$t^5\,b^{-1}\,t^{-1}b^{-1}\,t^{-1}b^{-1}\,t^{-1}b^{-1}\,t^{-1}b^{-1}\,t^{-1}b^5[s,b]u^3v^6 = 1$

i.e. $u^3v^6 = [t,b]^{15}[b,s]$ by Lemma 2.2.5.

Since $[t,b]^{15}$ is already a cube we have to consider substitutions of s to express [b,s] as a cube.

Let $s = zb^l \bmod G'$. Substituting this in we obtain as a general solution that l is arbitrary and z is a cube (i.e. let $z = q^3$).

Therefore $u^3v^6 = [t,b]^{15}[q,s]^3$.

So $u = [t,b]^5[q,s]$ and $v = 1$.

Therefore all solutions are given by

$x = t^2\ [t,b]^5[q,s]$, $y = q^3b^1r$ (where $r \in G'$), $z = t^{-1}b^{-1}$.

2.7.7:

Suppose we have the problem $[x,a][b,y][x,y] = 1$ in G, a free nilpotent group of class 2 generated by {a,b}.

Note that $[x,a][b,y][x,y] \in G'$. The problem is trivial in the abelian case (the most general solution is {x/x,y/y}. If we use the method of the previous examples (i.e. consider {xu/x,yv/y} where $u,v \in G'$) the u,v cancel out and we are left with the problem we started with. This is an example where h = 0 and we have to solve a τ-system of diophantine equations.

Let $x = a^eb^fr$ and $y = a^gb^hs$ (where $r,s \in G'$). Substituting this into the problem we obtain $[b,a]^f[b,a]^g[b,a]^{fg-he} = 1$.

Thus the solutions are obtained from the diophantine equation f + g + fg - he = 0.

2.7.8:

Suppose we have the problem $[a,b][a,x][a,y] = 1$ in a free nilpotent group of class 2.

Note again that we have a member of G' to be unified with 1.

Let $x = a^eb^fz\ r$ and $y = a^gb^ht\ s$ (where $r,s \in G'$). Substituting this into the problem we obtain $[a,b]^{f+h+1}[a,zt] = 1$.

Thus the solutions are obtained from the τ-system f+h+1 = 0 and the fact that zt = 1. Therefore all solutions are given by: $x = a^eb^fzr$ and $y = a^gb^{-f-1}z^{-1}\ s$ (where $r,s \in G'$).

In the next chapter we consider decidability problems in nilpotent groups with respect to unification. We also briefly consider the decidability of problems containing only one variable in nilpotent groups of class k as considered by Repin

[57]. As we have seen problems in nilpotent groups of class k and exponent p are always decidable because there is always a finite number of possible solutions to the kth power free diophantine equations of degree k that may occur.

Chapter 3: The Undecidability of the Unification Problem for Nilpotent Groups of Class ≥ 5

3.1 Introduction:

We are considering the unification problem for the theories of nilpotent groups of class ≥ 5. By Theorem 1.4.2 this is equivalent to the problem of constructing an algorithm which will solve any equation in a free nilpotent group of class ≥ 5. V.A. Romankov [60] in 1977 showed that the endomorphic reducibility problem is undecidable for free nilpotent groups of class ≥ 9. Hence the Unification Problem for the theories of nilpotent groups of class ≥ 9 is undecidable. We will reduce this number to 5. Our result will follow from the proof that there cannot exist an algorithm which will solve any equation in a free nilpotent group of class 5. This is established by reducing the problem to that of algorithmically solving an arbitrary diophantine equation of degree 4. The result then follows by appeal to the well known result that the solution of arbitrary diophantine equations of degree ≥ 4 is undecidable [50].

3.2 Preliminaries

The necessity of the reults we present in this section will become apparent in the proof of the main theorem (Theorem 3.3.2) of this chapter. Firstly, however, we will introduce a particular free nilpotent group of class 5 which we will call F.

3.2.1 Definition:

Let F be the free nilpotent group of class 5 on the two generators a_0, a_1. It follows from Definitions 2.2.3 that the basic commutators of weight 5 in F are $[a_1,a_0,a_0,a_0,a_0]$, $[a_1,a_0,a_0,a_0,a_1]$, $[a_1,a_0,a_0,a_1,a_1]$, $[a_1,a_0,a_1,a_1,a_1]$, $[[a_1,a_0,a_0],[a_1,a_0]]$, and $[[a_1,a_0,a_1],[a_1,a_0]]$. We will write these as b_1, b_2, b_3, b_4, b_5 and b_6 respectively. We already know by Lemma 2.2.4 that these basic commutators form a basis of the free abelian subgroup F_4.

The following lemmas which are applications of the collecting process allow us to isolate b_1 in a certain expression of F that occurs in the proof of Theorem 3.3.2.

Lemma 3.2.2:

Suppose we have the commutator $[a_1, a_0, a_i, a_j, a_k]$ in F_4 where $i,j,k \in \{0,1\}$. If we write this in terms of the basic commutators of Definition 3.2.1 then b_1 does not occur in the expression unless $i = j = k = 0$.

Proof:

Since b_1, b_2, b_3 and b_4 are linearly independent the only commutators we have to consider are:

(1) $[a_1, a_0, a_0, a_1, a_0]$

(2) $[a_1, a_0, a_1, a_0, a_0]$

(3) $[a_1, a_0, a_1, a_1, a_0]$

(4) $[a_1, a_0, a_1, a_0, a_1]$

We will prove the result for (1). The proofs for (2), (3) and (4) follow similarly.

To ease the computation of the collecting process defined in Lemma 2.2.5 and Remark 2.2.6 we will consider $[a_1, a_0, a_0, a_1, a_0]^{-1}$. In order to keep track of the proof the following terminology will be introduced. Suppose we have an expression $C = c_1 \ldots c_n$ where c_i (for $1 \leq i \leq n$) is a basic commutator of F. When discussing the basic commutator in the ith *position* of C we mean the commutator c_i.

We have $[a_1, a_0, a_0, a_1, a_0]^{-1}$

$= a_0^{-1}a_1^{-1}[a_1, a_0, a_0]^{-1}a_1[a_1, a_0, a_0]a_0[a_1, a_0, a_0, a_1]$.....(a)

by the definition of a commutator.

Now using the collecting process to move the a_0 in the sixth

position of expression (a) to the left to cancel out with the a_0^{-1} in the first position we obtain

$$[a_1,a_0]^{-1}a_1^{-1}[a_1,a_0,a_0,a_0]^{-1}[a_1,a_0,a_0]^{-1}a_1[a_1,a_0][a_1,a_0,a_0]$$

$$[a_1,a_0,a_0,a_0][a_1,a_0,a_0,a_1].....(b)$$

We now use the collecting process again to bring the a_1^{-1} in the second position of expression (a) to the first position. This gives us

$$a_1^{-1}[a_1,a_0,a_1][a_1,a_0,a_1,a_1,a_1][a_1,a_0,a_1,a_1]^{-1}[a_1,a_0]^{-1}$$

$$[a_1,a_0,a_0,a_0]^{-1}[a_1,a_0,a_0]^{-1}a_1[a_1,a_0][a_1,a_0,a_0][a_1,a_0,a_0,a_0]$$

$$[a_1,a_0,a_0,a_1].....(c)$$

we now use the same method to move the a_1 in the eighth position of expression (c) to the left to cancel out with the a_1 in the first position to obtain

$$[a_1,a_0]^{-1}[a_1,a_0,a_0,a_0,a_1]^{-1}[a_1,a_0,a_0,a_0]^{-1}[a_1,a_0,a_0,a_1]^{-1}$$

$$[a_1,a_0,a_0]^{-1}[a_1,a_0][a_1,a_0,a_0][a_1,a_0,a_0,a_0][a_1,a_0,a_0,a_1]...(d)$$

Once again we use the same method to move the $[a_1,a_0]$ in the sixth position of expression (d) to the left to cancel with the $[a_1,a_0]$ in the first position. The resulting expression involves only commutators of weight 3,4 or 5. We then use the fact (derived from Lemma 2.2.5) that in a nilpotent group of class 5 such commutators will commute with each other. Thus we obtain

$$[a_1,a_0,a_0,a_0,a_1]^{-1}[[a_1,a_0,a_0],[a_1,a_0]]^{-1}$$

Hence $[a_1,a_0,a_0,a_1,a_0] = b_2b_5$.

Lemma 3.2.3:

$[[a_1,a_0,a_0a_1],[a_1,a_0]] = [[a_1,a_0,a_0],[a_1,a_0]][[a_1,a_0,a_0],[a_1,a_0]]$

Proof:

We expand $[[a_1,a_0,a_0a_1],[a_1,a_0]]$ and apply the collecting process as in Lemma 3.2.2.

Now having considered the above preliminaries we can consider the main result of this chapter.

3.3 The undecidability of the unification problem for nilpotent groups of class ≥ 5

The work presented in this chapter rests upon the following theorem:

Theorem 3.3.1 (c.f. [50], [60]):

There is some diophantine polynomial D of degree 4 in a finite number of variables $x_1, \ldots, x_n$ and constants $c_1, \ldots, c_m$ such that the question of the existence of integer solutions of D = 0 (where D contains a parameter) is undecidable.

We now state and prove the main result of this chapter.

Theorem 3.3.2:

The Unification Problem for the theory of nilpotent groups of class 5 is undecidable.

Proof:

We will first show that there cannot exist an algorithm which will solve equations in the group F introduced in Definitions 3.2.1 above.

Consider D as given by Theorem 3.3.1. Then D can be written as a sum of monomials of the form;

c_r, $c_r x_{(i,1)}$, $c_r x_{(i,1)} x_{(i,2)}$, $c_r x_{(i,1)} x_{(i,2)} x_{(i,3)}$ and

$c_r x_{(i,1)} x_{(i,2)} x_{(i,3)} x_{(i,4)}$

where $(i,j) \in \{1, \ldots, n\}$ and $r \in \{1, \ldots, m\}$.

Let D_m be the set of all such monomials occurring in D and let $f: D_m \rightarrow F$ be the function defined as follows:

$f(c_r) = [a_1{}^{c_r}, a_0, a_0, a_0, a_0]$

$f(c_r x_{(i,1)}) = [a_1{}^{c_r}, a_0, a_0, a_0, y_{(i,1)}]$

$f(c_r x_{(i,1)} x_{(i,2)}) = [a_1{}^{c_r}, a_0, a_0, y_{(i,1)}, y_{(i,2)}]$

$f(c_r x_{(i,1)} x_{(i,2)} x_{(i,3)}) = [a_1{}^{c_r}, a_0, y_{(i,1)}, y_{(i,2)}, y_{(i,3)}]$

$f(c_r x_{(i,1)} x_{(i,2)} x_{(i,3)} x_{(i,4)}) = [a_1{}^{c_r}, y_{(i,1)}, y_{(i,2)}, y_{(i,3)}, y_{(i,4)}]$

where $(i,j) \in \{1, \ldots, n\}$ and $y_1, \ldots, y_n$ are new variables standing for elements of F.

Now consider the following equation in F:

$$[a_1, a_0, a_0, t_1, a_1][a_1, a_0, a_1, a_1, t_2][[a_1, a_0, t_3], [a_1, a_0]] \prod f(D_m) = 1$$

where t_1, t_2 and t_3 are new variables standing for elements of F but not occurring in $f(D_m)$.

Any solution in F will be given by $y_{(i,j)} = a_1{}^{z_{(i,j)}} a_0{}^{x_{(i,j)}} \bmod F_1$ and $t_p = a_1{}^{u_p} a_0{}^{v_p} \bmod F_1$ where u_p, v_p and $z_{(i,j)}$ are new integer variables and $p \in \{1,2,3\}$.

So substituting for $y_{(i,j)}$ and using Lemmas 2.2.7, 3.2.2 and 3.2.3 we have

$$[a_1, a_0, a_0, t_1, a_1][a_1, a_0, a_1, a_1, t_2][[a_1, a_0, t_3], [a_1, a_0]]$$

$$.[a_1, a_0, a_0, a_0, a_0]^D$$

$$.\prod\{[a_1, a_0, a_i, a_j, a_k]^{E(i,j,k)} : i,j,k \in \{0,1\}, \text{ not all } 0\} = 1$$

where $E(i,j,k)$ is a diophantine polynomial of degree 4 in $z_{(i,j)}$ and $x_{(i,j)}$.

By Lemma 2.2.4 and Lemma 3.2.2 we have

$$[a_1, a_0, a_0, t_1, a_1][a_1, a_0, a_1, a_1, t_2][[a_1, a_0, t_3], [a_1, a_0]\ b_1{}^D\ b_2{}^{E_1}\ b_3{}^{E_2}$$

$$b_4{}^{E_3}\ b_5{}^{E_4}\ b_6{}^{E_5} = 1$$

where E_1, E_2, E_3, E_4 and E_5 are diophantine polynomials of degree 4 in $z_{(i,j)}$ and $x_{(i,j)}$.

Now if this has a solution then D = 0 must have a solution because when the first three terms are expressed as a product of basic commutators of weight 5 they do not involve b_1 by Lemma 3.2.2.

Conversely, if D = 0 has a solution then the equation has a solution, namely $t_1 = a_0^{-E_1}a_1^{-E_2}$, $t_2 = a_1^{-E_3}$, $t_3 = a_0^{-E_4}a_1^{-E_5}$.

Hence by Theorem 3.3.1 we have a problem $w(a_0, a_1, y_1, \ldots, y_n) = 1$ in F which is recursively insoluble.

Now if we have a unification algorithm for N_5G then this will tell us whether we can solve w = 1 in N_5G and by Theorem 1.4.2 this is equivalent to being able to solve w = 1 in a free nilpotent group of class 5 on

$$\{a_0, a_1, a_2, \ldots\} \cup \{v_0, v_1, v_2, \ldots\}.$$

The members of $\{v_0, v_1, v_2, \ldots\}$ represent variables.

If we can solve w = 1 then the algorithm will produce a substitution

$$\theta = \{a_0^{s(0,i)} \ldots a_m^{s(m,i)} v_0^{t(0,i)} \ldots v_k^{t(k,i)} \bmod F_1/y_i : i = 1, \ldots, n\}$$

such that $w\theta = 1$.

Now $w\theta$ can be written as a product of the basic commutators of weight 5 generated by $\{a_0, a_1, a_2, \ldots\} \cup \{v_0, v_1, v_2, \ldots\}$. For every basic commutator c occurring in $w\theta$, c^{-1} must occur in $w\theta$. Now only a_0 and a_1 occur in w so we can replace the substitution by $\sigma = \{a_0^{s(0,i)} a_1^{s(1,i)} \bmod F_1/y_i : i = 1, \ldots, n\}$ and $w\sigma = 1$

because any basic commutator containing a_i (where $i > 1$) or v_i (where $i \geq 0$) simply cancels away.

Hence the Unification Problem is undecidable for nilpotent groups of class 5.

The following theorem extends this undecidability result to nilpotent groups of class ≥ 5.

Theorem 3.3.3:

The Unification Problem for the theory of nilpotent groups of class ≥ 5 is undecidable.

Proof:

Let J be the free nilpotent group of class k ($k > 5$) on the two generators a_0 and a_1.

As in Theorem 3.3.2 we will first show that there cannot exist an algorithm which will solve equations in J.

Consider D as given by Theorem 3.3.1. and written as a sum of monomials of the form given in the proof of Theorem 3.3.2

Let D_m be the set of all such monomials occurring in D and let $f:D_m \rightarrow F$ be the function defined as in the proof of Theorem 3.3.2.

Now let w_J be the word introduced in the proof of Theorem 3.3.2 but lying in J. For the moment let us consider the equation $w_J = 1$ in J.

Any solution in J will be given by

$$y_{(i,j)} = a_1^{z_{(i,j)}} a_0^{x_{(i,j)}} e_s \bmod J_s \text{ and } t_p = a_1^{u_p} a_0^{v_p} f_s \bmod J_s$$

where u_p, v_p and $z_{(i,j)}$ are new integer variables, $p \in \{1,2,3\}$, $s = k - 4$ and e_s, f_s are expressions involving all the basic

commutators of weight $\leq s$ and ≥ 2 on $\{a_0, a_1\}$ (each basic commutator in e_s, f_s will have as its power a new distinct integer variable).

So substituting for $y_{(i,j)}$ and using Lemmas 2.2.7, 2.2.4, 3.2.2 and 3.3.3 we have

$$[a_1,a_0,a_0,t_1,a_1][a_1,a_0,a_1,a_1,t_2][[a_1,a_0,t_3],[a_1,a_0]\ b_1^D\ b_2^{E_1}\ b_3^{E_2}$$

$$b_4^{E_3}\ b_5^{E_4}\ b_6^{E_5} = 1 \pmod{J_5} \ldots\ldots\ldots (1)$$

where E_1, E_2, E_3, E_4 and E_5 are diophantine polynomials of degree 4 in $z_{(i,j)}$ and $x_{(i,j)}$. Let $C(k)$ be the set of basic commutators of weight $\leq k$ and > 5 on $\{a_0, a_1\}$. Now $C(k)$ is finite so suppose $C(k) = \{c_0, .., c_q\}$. This means we have from (1):

$$[a_1,a_0,a_0,t_1,a_1][a_1,a_0,a_1,a_1,t_2][[a_1,a_0,t_3],[a_1,a_0]\ b_1^D\ b_2^{E_1}\ b_3^{E_2}$$

$$b_4^{E_3}\ b_5^{E_4}\ b_6^{E_5}\ c_0^{F_0} \ldots . c_q^{F_q} = 1$$

where $F_0, \ldots, F_q$ are diophantine polynomials in the integer variables that we have introduced. Now suppose we take each element c_i of $C(k)$ and let $c_i(r_i)$ be c_i with the first a_0 from the left replaced by a new distinct variable r_i ranging over J.

Now consider the problem $c_0(r_0) \ldots\ c_q(r_q) w_J = 1$

Now if this has a solution then $D = 0$ must have a solution because when the terms $[a_1,a_0,a_0,t_1,a_1]$, $[a_1,a_0,a_1,a_1,t_2]$ and $[[a_1,a_0,t_3],[a_1,a_0]]$ in w_J are expressed as a product of basic commutators of weight 5 they do not involve b_1. Clearly b_1 is not involved when the commutators of weight > 5 are written in terms of basic commutators of weight > 5.

Conversely, if D = 0 has a solution then the equation has a solution. We can see this by substituting $t_1 = a_0^{-E_1}a_1^{-E_2}$, $t_2 = a_1^{-E_3}$, $t_3 = a_0^{-E_4}a_1^{-E_5}$. This substitution will give us

$$c_0(r_0)\ldots\ c_q(r_q)b_1^{D}\ \ c_0^{F'_0}\ldots.c_q^{F'_q} = 1$$

F'_i may differ from F_i because in the manipulation which allowed us to cancel $b_2^{E_1}\ b_3^{E_2}\ b_4^{E_3}\ b_5^{E_4}\ b_6^{E_5}$ we may well have introduced some commutators of weight > 5 (In fact in most calculations we will have introduced them).

Now the substitution $r_i = a_0^{-F'_i}$ will leave us with $b_1^{D} = 1$.

Hence by Theorem 3.5 we have a problem $w(a_0, a_1, y_1, \ldots, y_n) = 1$ in J which is recursively insoluble.

Now we know that the Unification Problem for nilpotent groups of class k is equivalent to being able to solve w = 1 in a free nilpotent group of class k on

$$\{a_0, a_1, a_2, \ldots\}\cup\{v_0, v_1, v_2, \ldots\}$$

where the members of $\{v_0, v_1, v_2, \ldots\}$ represent variables. By a very similar argument to that employed in the proof of Theorem 3.3.2 we can consider substitutions involving only a_0 and a_1. Hence the Unification Problem is undecidable for nilpotent groups of class ≥ 5.

3.3.4 Remark:

We gave a proof of this undecidability result by constructing a problem in a free nilpotent group of class 5 which reduced to the problem of solving an arbitrary diophantine equation of degree 4. We then appealed to Theorem 3.3.1. This showed the result holds for class 5. We then considered a very similar

problem lying in a free nilpotent group of higher nilpotency class and established the result for class > 5.

An alternative proof would have been to construct a problem in a nilpotent group of class k which reduces to the problem of solving an arbitrary diophantine equation of degree k-1. The proof of Theorem 3.3.3 would then follow from Theorem 3.3.1 and the fact that an arbitrary diophantine equation of degree > 4 is equivalent to one of degree 4 (see [60]).

We now briefly discuss the work done by Repin [57] on the undecidability of the unification problem for nilpotent groups when considering problems containing only one variable.

3.4 Problems in one variable only

In 2.5 we presented a unification algorithm for nilpotent groups of class 2 with respect to problems containing only one variable. As was seen in Remark 2.5.9 matters were simplified considerably because when we restrict the unification problem in this way the troublesome diophantine equations of degree 2 do not occur. So in this particular case we do not have a relative algorithm but an outright one.

However, we also saw in Remark 2.4.8 that the restriction to problems in just one variable for nilpotent groups of class k ($k \geq 3$) means that we have to solve diophantine equations of degree $k - 1$. So although we can construct an outright algorithm for this restricted class of problems for nilpotency class 2 the unification algorithm for nilpotent groups of class k is relative to solving certain diophantine equations of degree $k - 1$. This would suggest that the unification problem is undecidable for nilpotent groups of class 5 when restricting to one variable (as we showed it to be for an arbitrary number of variables in Theorem 3.3.2). However, the method of proof employed in Theorem 3.3.2 does not work when we restrict to one variable. In the proof of this theorem we coded up an arbitrary diophantine equation of degree 4 by associating a diophantine monomial $c_r x_{(i,1)} x_{(i,2)} x_{(i,3)} x_{(i,4)}$ in four variables with the commutator $[a_1{}^{c_r}, y_{(i,1)}, y_{(i,2)}, y_{(i,3)}, y_{(i,4)}]$ in four variables. The point is that we need the four variables $y_{(i,1)}, y_{(i,2)}, y_{(i,3)}, y_{(i,4)}$ in the group equation to encode the four variables $x_{(i,1)} x_{(i,2)} x_{(i,3)} x_{(i,4)}$ of the arbitrary diophantine equation. If we restrict to one variable only for problems in a free nilpotent group of class 5 then we can encode

certain diophantine equations of degree 4 but we cannot encode an arbitrary diophantine equation of degree 4. In fact using the same method as in the proof of Theorem 3.3.2 the class of diophantine equations of degree 4 that we obtain are of the form $ax^4 + bx^3 + cx^2 + dx + e = 0$ where x is an integer variable and a,b,c,d,e are integer constants. If the problem of algorithmically solving any member of this class of diophantine equations was undecidable then we could establish the undecidability of the unification problem for nilpotent groups of class 5 when restricting to the use of one variable only. Unfortunately the algorithmic solvability of this class of diophantine equations is decidable. So to establish an undecidability result for this restricted class of problems we have to turn away from the methods already developed and consider something new.

Repin's result [57] is that for $k \geq 10^{20}$ there is no algorithmic way of solving equations containing just one variable in a free nilpotent group of class k. The proof of this result rests upon a theorem of number theory as do Romankov's results (in [60] and [61]) and as do the proofs of Theorems 3.3.2 and 3.3.3. Repin, however, does not use the Matiyasevitch theorem [50]. He uses a theorem presented by J.P.Jones [30] which says that there exists a class M of diophantine polynomials of degree 2.10^5 containing fourteen variables such that given $p \in M$ the problem of algorithmically finding a solution to $p = 0$ is undecidable. As we have seen when we consider unification for nilpotent groups there is a direct correlation between the degree of the diophantine equations that occur and the nilpotency class of the group. The dependency upon diophantine equations of degree 2.10^5 results in obtaining the

undecidability theorem for nilpotency class $k > 10^{20}$.

A research problem that has been left open and that is prompted by Remark 2.5.11 is to show that the unification problem when considering one variable only is undecidable for nilpotent groups of class k for some $k < 10^{20}$.

Chapter 4: Equation Solving in Partially Commutative Monoids

4.1 Introduction:

The manipulation of strings is an important tool in many areas of computer science. This is the main reason why unification for the theory of semi-groups, and abelian semi-groups in particular, has attracted such attention. The aim of this chapter is to consider equation solving with respect to partially commutative monoids (semigroups with an identity).

In the title of the thesis we referred to `nilpotent monoids' since the structures we study are a natural analogue of those considered in Chapter 2. However, the word "nilpotent" is generally not applied to monoids since it implies the use of quotient structures. We therefore adopt the less contentious term "partially commutative monoid" to capture what we have in mind.

We define a class $\mathcal{N}$ of partially commutative monoids that we shall present and study in some detail. We also define another class $\mathcal{M}$ of partially commutative monoids whose members have less relations than the members of $\mathcal{N}$. We will show that there is an algorithm to solve equations in the structures belonging to $\mathcal{N}$ relative to having an algorithm which will solve certain systems of quadratic diophantine equations over the positive integers, which we will call ρ-systems (we will begin this chapter by taking a close look at these ρ-systems). We also outline the difficulties involved when solving equations in $\mathcal{M}$.

4.2 Preliminary Definitions:

4.2.1 Definitions of ρ-polynomials and ρ-systems

We now introduce the certain special types of polynomials and equations mentioned above that arise naturally when considering unification and equation solving in partially commutative monoids. These correspond to the τ-equations and τ-systems that occurred when we considered unification for nilpotent groups of class 2 in Chapter 2. Their use in solving equations in partially commutative monoids will be illustrated in 4.3.

Let X and Y be disjoint finite sets of distinct non-negative integer variables. A *ρ-polynomial* is a polynomial p of the form

$$p = K + \sum_{i=1}^{m} c_i y_i + \sum_{j=1}^{n} d_j x_j + \sum_{1 \le i < j \le n} \varepsilon_{(i,j)} x_i x_j$$

where (1) $y_i \in Y$, $x_j \in X$

(2) $c_i, d_j, \varepsilon_{(i,j)} \in Z$ and not all c_i are zero.

We shall consider finite sets $P = \{p_1, \ldots, p_t\}$ of ρ-polynomials where

$$p_s = K_s + \sum_{i=1}^{m} c_{(s,i)} y_{(s,i)} + \sum_{j=1}^{n} d_{(s,j)} x_{(s,j)} + \sum_{1 \le i < j \le n} \varepsilon_{(i,j)} x_{(s,i)} x_{(s,j)}$$

We say that a set of ρ-polynomials P is a *valid* set if no $y_{(s,i)}$ occurs in more than one member of P. Let P be a valid set of ρ-polynomials and let us associate with each $p_s \in P$ a pair of distinct linear diophantine polynomials $L_s = \{L_{(s,1)}, L_{(s,2)}\}$ where $L_{(s,1)}$ is of the form

$$e_{(s,1)} x_{(s,1)} + \ldots + e_{(s,q)} x_{(s,q)}$$

and $L_{(s,2)}$ is of the form

$$e_{(s,q+1)}x_{(s,q+1)} + \ldots + e_{(s,n)}x_{(s,n)}$$

where $e_{(s,1)}, \ldots, e_{(s,n)}$ are integer constants and if a variable occurs in $L_{(s,i)}$ $(i = 1,2)$ it does not occur in any other member of $L_1 \cup L_2 \cup \ldots \cup L_t$.

Note: For $r \neq s$ the elements of L_r and L_s need not be distinct. Indeed in the cases we require later on (for $t > 1$) there will always exist L_r and L_s $(r \neq s)$ such that $|L_r \cup L_s| < 4$.

Now let us associate with each $p_s \in P$ another ρ-polynomial P'_s of the form

$$p'_s = K'_s + \sum_{i=1}^{m} c'_{(s,i)}y'_{(s,i)} + \sum_{j=1}^{n} d'_{(s,j)}x_{(s,j)} + \sum_{1\leq i<j\leq n} \varepsilon'_{(i,j)}x_{(s,i)}x_{(s,j)}$$

where (1) $y'_{(s,1)}, \ldots, y'_{(s,m)}$ are distinct from $y_{(s,1)}, \ldots, y_{(s,m)}$ and any variable occurring in P'_r for $r \neq s$.

(2) $K'_s, c'_{(s,1)}, \ldots, c'_{(s,m)}, d'_{(s,1)}, \ldots, d'_{(s,n)}, \varepsilon'$ may or may not be the same as $K_s, c_{(s,1)}, \ldots, c_{(s,m)}, d_{(s,1)}, \ldots, d_{(s,n)}, \varepsilon$.

(3) The variables $x_{(s,1)}, \ldots, x_{(s,n)}$ are the same as those that appear in P_s.

Note: P' is a valid set of ρ-polynomials.

We define a *ρ-system* to be a system of diophantine equations consisting of some $\{P_1 - P'_1 = 0, \ldots, P_t - P'_t = 0\}$ (where $P = \{P_i: 1 \leq i \leq t\}$ and $P' = \{P_i': 1 \leq i \leq t\}$ are valid) together with the set of linear diophantine equations formed by putting each element of some corresponding $L_1 \cup L_2 \cup \ldots \cup L_t$ equal to zero.

Later on in this chapter we will be assuming that we have an algorithm to find all the solutions (over the positive integers) of any given ρ-system. A ρ-system consists of linear diophantine equations and square free diophantine equations of degree 2. The algorithm for solving systems of equations in the members of $\mathcal{N}$ presented in 4.3 could simply be presented as relative to the algorithmic solution of such systems of diophantine equations. However, the diophantine equation algorithm used would actually solve a larger class of systems of diophantine equations than is required. Obviously the more specialised the class of systems of diophantine equations considered the greater the chance of constructing an algorithm to solve them. It is for this reason that we have presented the definition of ρ-systems in such detail. The ρ-systems presented above are precisely the systems of diophantine equations that occur when considering equation solving in the members of $\mathcal{N}$.

4.2.2 The structures we will be working with:

Now we will introduce certain structures that we will be attempting to solve equations in. These structures are monoids (i.e. semigroups with identity). First, however, let us define the notion of a block. Let $A = \{a_i : 1 \leq i \leq n\}$ and suppose we have a word over A of the form:

$$w = a_{(p,1)}{}^{g_1} a_{(p,2)}{}^{g_2} \ldots a_{(p,k)}{}^{g_k}$$

where $(p,i) \in \{1,..,n\}$, g_i is a non-negative integer and $(p,i) \neq (p,i+1)$ for $i = 1,..,k$. We will call each subword $a_{(p,i)}{}^{g_i}$ of w a *block*. We now present a formal definition of the structures that we will be working in.

4.2.2.1 Definition of the structures belonging to $\mathcal{N}$:

Let $A = \{a_i: 1 \leq i \leq n\}$, $B = \{b_{(i,j)}: 1 \leq i < j \leq n\}$ and $B' = \{b'_{(i,j)}: 1 \leq i < j \leq n\}$ be disjoint sets of generators and let $\mathcal{N}$ be the set of all monoids given by a finite presentation of the form:

$$N = < \quad A \cup B \cup B' : \; b_{(i,j)}x = xb_{(i,j)} \quad (x \in A \cup B \cup B')$$

$$b'_{(i,j)}x = xb'_{(i,j)} \quad (x \in A \cup B \cup B')$$

$$a_ia_j = a_ja_ib_{(i,j)} \quad (i < j)$$

$$a_ia_jb'_{(i,j)} = a_ja_i \quad (i < j) \quad >$$

To define a nilpotent structure we have to use the concept of a quotient structure, which we cannot use when considering monoids. The motivation for looking at the monoids defined above is that they have nilpotent like character.

4.2.2.2 Definition of the structures belonging to $\mathcal{M}$:

For this class of structures we will be considering the disjoint sets of generators $A = \{a_i: 1 \leq i \leq n\}$ and $B = \{b_{(i,j)}: 1 \leq i < j \leq n\}$. We let $\mathcal{M}$ be the set of all monoids given by a finite presentation of the form:

$$M = < \quad A \cup B : \quad b_{(i,j)}x = xb_{(i,j)} \quad (x \in A \cup B)$$

$$a_ia_j = a_ja_ib_{(i,j)} \quad (i < j, \; (i,j) \in Y)$$

$$a_ia_jb_{(i,j)} = a_ja_i \quad (i < j, \; (i,j) \notin Y) \quad >$$

where Y is any fixed subset of $\{(i,j): 1 \leq i < j \leq n\}$

The motivation for studying $\mathcal{M}$ is that its members are

partially commutative monoids which are, in a certain sense, "more free" than the members of $\mathcal{N}$. That is each member of $\mathcal{M}$ has a corresponding member of $\mathcal{N}$ with "more" relations. However, we will show that the format of the substitutions that are required strongly suggest that the equation solving problem for the members of $\mathcal{M}$ is not decidable.

Before we consider these structures, however, we will first turn our attention to the problem of constructing an algorithm to solve equations in the structures defined in 4.2.2.1.

4.3 Equation solving in N ∈ $\mathcal{N}$:

Now before we consider solving equations in N ∈ $\mathcal{N}$ let us first prove the following Lemma about words lying in N:

Lemma 4.3.1:

Let w be a word over A ∪ B ∪ B'. Then w can be written in N in the form: $w = a_1^{u_1}\ldots a_n^{u_n}\ \Pi\{b_{(i,j)}^{v_{(i,j)}} : b_{(i,j)} \in B\}$

$$\Pi\{b'^{v'_{(i,j)}}_{(i,j)} : b'_{(i,j)} \in B'\}$$

where $u_i, v_{(i,j)}$ and $v'_{(i,j)}$ are non-negative integers.

Proof:

Suppose we have w in the form

$$w = a_1^{f_1}\ldots a_n^{f_n}\ a_i^{g_i}a_j^{g_j}\ldots a_k^{g_k}\ \Pi\{b_{(i,j)}^{v_{(i,j)}} : b_{(i,j)} \in B\}.$$

$$\Pi\{b'^{v'_{(i,j)}}_{(i,j)} : b'_{(i,j)} \in B'\}$$

where f_i and g_1 are non-negative integers, $a_1^{f_1}\ldots a_n^{f_n}$ is ordered and $a_i^{g_i}a_j^{g_j}\ldots a_k^{g_k}$ is unordered. We prove the result by induction on the number of blocks of $a_i^{g_i}a_j^{g_j}\ldots a_k^{g_k}$.

The base case is trivial.

For the induction step we first note that in the monoid N we have the following relations:

$$a_i^{g_i}a_n^{f_n} = a_n^{f_n}a_i^{g_i}b_{(i,n)}^{f_n g_i} \quad \ldots\ldots \ (\alpha)$$

$$a_i^{g_i}a_n^{f_n}b'^{f_n g_i}_{(i,n)} = a_n^{f_n}a_i^{g_i} \quad \ldots \quad (\beta)$$

(we know that i < n so we know that the corresponding elements of B, B' are $b_{(i,n)}$, $b'_{(i,n)}$ and *not* $b_{(n,i)}$, $b'_{(n,i)}$ which will not even exist in N)

Now we have the choice of using either (α) or (β) to move $a_i^{g_i}$

past $a_n{}^{f_n}$. We can, however, only use (α) if $b_{(i,n)}$ occurs in w with power at least $f_n g_i$ because otherwise we would introduce a negative power into a monoid. It is always possible to use (β). In general any algorithm which writes w in the required form will at this point be faced with deciding which of the two relations to use. We can, however, write the word in a canonical form by always using the relation which takes away a $b_{(i,n)}$ if there are any occurrences of $b_{(i,n)}$ to take away. If not we use the relation which introduces a $b'_{(i,n)}$.

Having moved $a_i{}^{g_i}$ past $a_n{}^{f_n}$ we can now move it past $a_{n-1}{}^{f_{n-1}}$ and so on until it is in the correct position in the ordered subword $a_1{}^{f_1}\ldots a_n{}^{f_n}$. Since $a_j{}^{g_j}\ldots a_k{}^{g_k}$ is shorter than $a_i{}^{g_i}a_j{}^{g_j}\ldots a_k{}^{g_k}$ the result then follows by induction.

Note: As we are only making moves to the left we cannot go into an infinite loop.

4.3.2 Definition:

We are attempting to solve equations in any $N \in \mathcal{N}$. For every variable x in a particular equation we will be applying the substitution:

$$x = a_1{}^{x_1}\ldots a_n{}^{x_n}\,\Pi\{b_{(i,j)}{}^{y_{(i,j)}} : b_{(i,j)} \in B\}$$

$$.\Pi\{b'_{(i,j)}{}^{y'_{(i,j)}} : b'_{(i,j)} \in B'\}$$

where x_i, $y_{(i,j)}$ and $y'_{(i,j)}$ are non-negative integer variables i.e. our substitutions will have the format suggested by Lemma 4.3.1. This prompts the following definition and lemma:

Let V be a word over the alphabet

$A \cup B \cup B' \cup \{a_i{}^{t_j}: t_j$ is a non-negative integer variable$\}$

$\cup \{b_{(i,j)}{}^{q_k}: q_k$ is a non-negative integer variable$\}$

$\cup \{b'_{(i,j)}{}^{q'_k}: q'_k$ is a non-negative integer variable$\}$.

When non-negative integers are assigned to the distinct variables t_j, q_k, q'_k V then becomes a word in $A \cup B \cup B'$. Now because $B \cup B'$ is central V can be written in the form

$$Y(a_1,\ldots,a_n)\ \Pi\{b_{(i,j)}{}^{p_{(i,j)}}: b_{(i,j)} \in B\}$$

$$.\Pi\{b'_{(i,j)}{}^{p'_{(i,j)}}: b'_{(i,j)} \in B'\}$$

where $Y(a_1,\ldots,a_n)$ is a word over $A \cup \{a_i{}^{t_j}: t_j$ is a non-negative integer variable$\}$ and $p_{(i,j)}, p'_{(i,j)}$ are non-negative linear diophantine polynomials in the q's.

Lemma 4.3.3:

V can be written in N in the form:

$$V = a_1{}^{r_1}\ldots a_n{}^{r_n}\ \Pi\{b_{(i,j)}{}^{s_{(i,j)}}: b_{(i,j)} \in B\}$$

$$.\Pi\{b'_{(i,j)}{}^{s'_{(i,j)}}: b'_{(i,j)} \in B'\}$$

where (1) r_i is a unique non-negative linear diophantine polynomial over the non-negative integers.

(2) $s_{(i,j)}$ is a ρ-polynomial.

(3) $s'_{(i,j)}$ is a ρ-polynomial.

Proof:

Suppose we have V in the form

$$V = a_1{}^{d_1}\ldots a_n{}^{d_n}\ a_i{}^{e_i}a_j{}^{e_j}\ldots a_k{}^{e_k}\ \Pi\{b_{(i,j)}{}^{s_{(i,j)}}: b_{(i,j)} \in B\}.$$

$$\Pi\{b'_{(i,j)}{}^{s'_{(i,j)}}: b'_{(i,j)} \in B'\}$$

where $a_1{}^{d_1}\ldots a_n{}^{d_n}$ has already been ordered and $a_i{}^{e_i}a_j{}^{e_j}\ldots a_k{}^{e_k}$ is

unordered. The d's and the e's are either non-negative integers or non-negative integer variables with non-negative integer coefficients. In a similar way to the proof of Lemma 4.3.1 we can show that we can write the word in the intended order by induction on the number of blocks of $a_i^{e_i}a_j^{e_j}\ldots a_k^{e_k}$. For the induction step in this case we use the relations:

$$a_i^{e_i}a_n^{d_n} = a_n^{d_n}a_i^{e_i}b_{(i,n)}^{d_n e_i} \quad \ldots\ldots \quad (\alpha)$$

$$a_i^{e_i}a_n^{d_n}b'_{(i,n)}^{d_n e_i} = a_n^{d_n}a_i^{e_i} \quad \ldots\ldots \quad (\beta)$$

Now as before we can use either (α) or (β) to move $a_i^{g_i}$ past $a_n^{f_n}$ and we can only use (α) if we have a sufficiently large power of $b_{(i,n)}$. This means that when we come to substitute values for the variables of $s_{(i,n)}$ the result cannot be negative. We can, however write the word in the canonical form introduced in the proof of Lemma 4.3.1.

Now clearly (1) r_i is a unique non-negative linear diophantine polynomial over the non-negative integers.

Also (2) $s_{(i,j)}$ is a ρ-polynomial.

We show this by induction on the number l of moves made. Let $s_{(i,j)}(0) = p_{(i,j)}$ and $s_{(i,j)}(l)$ be the power of $b_{(i,j)}$ after l moves.

We know that $p_{(i,j)}$ is a linear non-negative diophantine polynomial over the non-negative integers (it is in fact the $\sum_{i=1} c_i y_i$ of Definition 4.2.1 together with some constant). Hence the base case is trivially true.

For the induction step suppose that $s_{(i,j)}(l)$ is a ρ-polynomial

and that move l+1 involves moving $a_i^{e_i}$ past $a_k^{e_k}$. Without loss of generality assume $i < k$.

We can use

$$a_i^{e_i}a_k^{e_k} = a_k^{e_k}a_i^{e_i}b_{(i,k)}^{e_k e_i} \quad \ldots\ldots \quad (\alpha_{l+1})$$

$$\text{or } a_i^{e_i}a_k^{e_k}b'^{e_k e_i}_{(i,k)} = a_k^{e_k}a_i^{e_i} \quad \ldots\ldots \quad (\beta_{l+1})$$

(the subscript in (α_{l+1}) and (β_{l+1}) is introduced to indicate the number of moves made)

If we use (β_{l+1}) then $s_{(i,j)}(l+1) = s_{(i,j)}(l)$.

If we use (α_{l+1}) then $s_{(i,j)}(l+1) = s_{(i,j)}(l) + e_i e_k$

Now e_i and e_k are either non-negative integers or non-negative integer variables (distinct from those occurring in $p_{(i,j)}$) with non-negative integer coefficients.

i.e. $e_i e_k$ is either (a) a constant

or (b) a variable with constant coefficient

(this would be part of the $\sum_{j=1}^{n} d_j x_j$ of Definitions 4.2.1)

or (c) a product of two distinct variables with constant coefficient (this would be a part of the $\sum_{1 \le i < j \le n} \varepsilon_{(i,j)} x_i x_j$

of Definitions 4.3.1).

Hence by induction the result holds.

Note: $s_{(i,j)}$ is not a unique ρ-polynomial because we are writing the word in a certain way, the canonical way defined in the proof of Lemma 4.3.1. Writing the word using a different choice of relation would result in a different ρ-polynomial.

The proof that (3) $s'_{(i,j)}$ is a ρ-polynomial follows by a

similar argument to the one used for $s_{(i,j)}$. It is not unique for the same reasons.

Now having the necessary preliminary results at our disposal we can look at an algorithm to solve any given equation in N.

4.3.4 An algorithm to solve equations in $N \in \mathcal{N}$

Suppose we are given the problem $w_1 = w_2$? in $N \in \mathcal{N}$. Now w_1, w_2 can be written in the following forms since $B \cup B'$ is central:

$$w_1 = u_1(a_1, \ldots, a_n, x_1, \ldots, x_m).$$

$$\Pi_1\{b_{(i,j)} : b_{(i,j)} \in B\}\ \Pi_1\{b'_{(i,j)} : b'_{(i,j)} \in B'\}$$

$$\text{and } w_1 = u_2(a_1, \ldots, a_n, x_1, \ldots, x_m).$$

$$\Pi_2\{b_{(i,j)} : b_{(i,j)} \in B\}\ \Pi_2\{b'_{(i,j)} : b'_{(i,j)} \in B'\}$$

where $x_1, \ldots, x_m$ are variables standing for any word in N.

Now for each variable x_k occurring in the problem we let

$$x_k = a_1^{x_{(1,k)}} \ldots a_n^{x_{(n,k)}}\ \Pi\{b_{(i,j)}^{y_{(i,j,k)}} : b_{(i,j)} \in B\}.$$

$$\Pi\{b'^{y'_{(i,j,k)}}_{(i,j)} : b'_{(i,j)} \in B'\}$$

where $x_{(i,k)}$, $y_{(i,j,k)}$, $y'_{(i,j,k)}$ are distinct non-negative integer variables. We can do this because we know by Lemma 4.3.1 that every word in N can be written in this form.

Having substituted these values of x_k (for $k = 1, \ldots, m$) we write the left hand side of the equation in the form:

$$a_1^{r_1} \ldots a_n^{r_n}\ \Pi_3\{b_{(i,j)}^{s_{(i,j)}} : b_{(i,j)} \in B\}$$

$$\Pi_3\{b'^{s'_{(i,j)}}_{(i,j)} : b'_{(i,j)} \in B'\}$$

by Lemma 4.3.3. In applying this Lemma we use the canonical

method of rewriting the word introduced in the proof of Lemma 4.3.1

We can also write the right hand side of the equation in the form:

$$a_1^{t_1} \ldots a_n^{t_n} \Pi_4\{b_{(i,j)}^{q_{(i,j)}} : b_{(i,j)} \in B\}$$

$$\Pi_4\{b'^{q'}_{(i,j)} : b'_{(i,j)} \in B'\}$$

by using the same canonical method.

If we now equate the powers of the independent generators and use parts (1),(2),(3) of Lemma 4.3.3 we see that we have derived the following ρ-system:

$$\{r_i - t_i = 0\} \cup \{s_{(i,j)} - q_{(i,j)} - (s'_{(i,j)} - q'_{(i,j)}) = 0\}$$

Note: $\{r_i - t_i\} = L_1 \cup L_2 \cup \ldots \cup L_t$, $\{s_{(i,j)} - q_{(i,j)}\} = P$ and $\{s'_{(i,j)} - q'_{(i,j)}\} = P'$ where $L_1 \cup L_2 \cup \ldots \cup L_t$, P and P' are the notations used in Definitions 4.2.1. It is easy to check that $\{s_{(i,j)} - q_{(i,j)}\}$ and $\{s'_{(i,j)} - q'_{(i,j)}\}$ are valid.

Now we are only interested in positive solutions to these ρ-systems because we are working in monoids. The following theorem tells us that if we have an algorithm to find all the solutions (over the positive integers) of any ρ-system then we have one to solve the given problem $w_1 = w_2$?:

Theorem 4.3.5:

We have a (positive) solution to this ρ-system if and only if we have a solution to the problem $w_1 = w_2$?

Proof:

The implication from left to right is obvious.

Conversely assume that we have any solution

$$\sigma = \{s_k(a_1,\cdot..a_n)\Pi\{b_{(i,j)}{}^{p_{(i,j,k)}}: b_{(i,j)} \in B\}$$

$$\Pi\{b'_{(i,j)}{}^{p'_{(i,j,k)}}: b'_{(i,j)} \in B'\}/x_k: k = 1,..,n\}$$

to the problem $w_1 = w_2$?

By Lemma 4.3.1 we can write this in the form:

$$\sigma = \{a_1{}^{u_{(1,k)}}\cdot..a_n{}^{u_{(n,k)}}\Pi\{b_{(i,j)}{}^{v_{(i,j,k)}}: b_{(i,j)} \in B\}$$

$$\Pi\{b'_{(i,j)}{}^{v'_{(i,j,k)}}: b'_{(i,j)} \in B'\}/x_k: k = 1,..,n\}$$

Now let us consider $w_1\sigma$ and apply Lemma 4.3.1 (using the canonical method defined there). We do the same for $w_2\sigma$.

Since σ is solution of $w_1 = w_2$? we have that

$x_{(i,k)} = u_{(i,k)}$, $y_{(i,j,k)} = v_{(i,j,k)}$, $y'_{(i,j,k)} = v'_{(i,j,k)}$ is a solution to the ρ-system.

We can summarise the results of this section so far in the following theorem:

Theorem 4.3.6:

If there is an algorithm to solve (over the positive integers) any ρ-system then we have an algorithm to solve any equation in $N \in \mathcal{N}$.

Proof:

The proof follows immediately from algorithm 4.3.4 and Theorem 4.3.5.

A natural question arising at this point is whether or not the converse of Theorem 4.3.6 is true i.e. is the solving of ρ-systems equivalent to the solving of equations in $N \in \mathcal{N}$? The following example shows that there is no direct way to derive equations in $\mathcal{N}$ from ρ-systems. The example also provides an

illustration of the equation solving method introduced above.

Example 4.3.7:

We showed in Algorithm 4.3.4 and Theorem 4.3.5 that for every equation in N there is a corresponding set of ρ-systems.

Suppose we have the monoid

$$L = \langle a,b,c,c' : ac = ca,\ bc = cb,$$

$$ac' = c'a,\ bc' = c'b,$$

$$ab = bac,\ abc' = ba \rangle \in \mathcal{N}.$$

Now consider the problem axb = ya ? in L. If we let

$$x = a^{x_a}b^{x_b}c^{x_c}c'^{x_{c'}} \text{ and } y = a^{y_a}b^{y_b}c^{y_c}c'^{y_{c'}}$$

(where $x_a, x_b, x_c, x_{c'}, y_a, y_b, y_c, y_{c'}$ are non-negative integer variables) we obtain

$$a^{x_a+1}\, b^{x_b+1}\, c^{x_c}\, c'^{x_{c'}} = a^{y_a}\, b^{y_b}\, a\, c^{y_c}\, c'^{y_{c'}}$$

This means that we can derive one of the following two ρ-systems:

$$\{x_a - y_a = 0,\ x_b - y_b + 1 = 0,\ x_c + y_b - y_c = 0,\ x_{c'} - y_{c'} = 0\}$$

$$\{x_a - y_a = 0,\ x_b - y_b + 1 = 0,\ x_c - y_c = 0,\ x_{c'} - y_b - y_{c'} = 0\}$$

Clearly if we were given either of these two ρ-systems we could work *backwards* and find the corresponding problem in L. However, this ρ-system can be slightly altered to create a ρ-system that has no corresponding equation in N.

Suppose that we were given the ρ-system:

$$\{2x_a - y_a = 0,\ x_b - y_b + 1 = 0,\ x_c + y_b - y_c = 0,\ x_{c'} - y_{c'} = 0\}.$$

Working backwards we would derive in L:

$$a^{x_a+1}xb = ya\ ?$$

Now if we had $b^{x_b}, c^{x_c}, c'^{x_{c'}}$ occuring in this equation then we could present it as an equation of the monoid by replacing

$a^x_a b^x_b c^x_c c'^x_{c'}$ with the variable x. However, we have only a^x_a occurring on its own. Because a variable standing for any element of L must involve more than a power of the single generator a there is no way to mop up the extra a^x_a. Hence the ρ-system does not have a corresponding problem in L (or any other member of $\mathcal{N}$).

Now having shown that equation solving in $N \in \mathcal{N}$ is dependent upon solving ρ-systems let us consider the structures defined in 4.2.2.2.

4.4 Equation solving in M ∈ $\mathfrak{M}$:

We will show that the format of the substitutions required suggests that the equation solving problem in the members of $\mathfrak{M}$ is not decidable. However, the results presented in 4.3 emerged from first looking at these structures.

Suppose we have N ∈ $\mathfrak{N}$ as defined in 4.2.2.1. In the proof of Lemma 4.3.1 we gave a method for writing any word w of N in a certain form which used the given relations of N to collect all the generators together. Given any two generators a_i and a_j of N $(i < j)$ there is always a corresponding $b_{(i,j)} \in B$ together with a relation $a_1a_2 = a_2a_1b_{(i,j)}$ and there is always a corresponding $b'_{(i,j)} \in B'$ together with a relation $a_1a_2b'_{(i,j)} = a_2a_1$. This means that given a subword a_2a_1 of w we can always move the a_1 past the a_2 using one of the given relations regardless of whether or not the appropriate members of $B \cup B'$ occur in w.

However, a corresponding member of $\mathfrak{M}$ contains the set B instead of $B \cup B'$ and will contain the relation $a_1a_2 = a_2a_1b_{(i,j)}$ or the relation $a_1a_2b_{(i,j)} = a_2a_1$ but not both. Thus given a subword a_2a_1 of w and the given relation we can always move the a_1 past the a_2 using that relation provided we have an occurrence of $b_{(i,j)}$ in w. There is of course no guarantee that we always have such an occurrence. To highlight the point we will construct two examples of words in certain members of $\mathfrak{M}$ which cannot be rewritten in the desired form. The first will be a simple case and for this case we will show a way round the problem. We will see that there is no way round the second

problem. For every member of $\mathfrak{M}$ there is a corresponding member of $\mathfrak{N}$. We will compare the problems we look at in, particular members of $\mathfrak{M}$, with the same problems in the corresponding member of $\mathfrak{N}$.

Example 4.4.1:

Suppose we are given the monoid

$$M = \langle a,b,c: ab = bac,\ ac = ca,\ bc = cb \rangle.$$

It is easy to check that $M \in \mathfrak{M}$ (comparing this with Definition 4.2.2.2 we have that $A = \{a,b\}$ and $B = \{c\}$).

Now suppose that we are given the word $abababc^3$ and that we wish to rewrite it in the form $a^pb^qc^r$ (as we did for the members of $\mathfrak{N}$ in 4.3). Then using the relations of M we can rewrite this as a^3b^3.

However, if we are given the word abagab then there is no way in M that we can write the word in the desired form.

The corresponding monoid in $\mathfrak{N}$ is

$$\langle a,b,c,c': ab = bac,\ abc' = ba,\ cx = xc \quad (x \in \{a,b,c,c'\}),$$
$$c'x = xc' \quad (x \in \{a,b,c,c'\}) \rangle.$$

Comparing this with Definition 4.2.2.1 we have that $A = \{a,b\}$, $B = \{c\}$ and $B' = \{c'\}$.

We can see that in this monoid the word abagab can be rewritten as $a^3b^3c'^3$.

We could, however, get round the problem in M by changing the order in which we wish to rewrite the generators. That is we could always rewrite any word in M in the form $b^pa^qc^r$. This provides us with the following Lemma:

Lemma 4.4.2:

If we have an algorithm to solve any ρ-system then we have an algorithm to solve any equation in a three generator element of $\mathfrak{M}$.

Proof:

There are just 2 three generator elements of $\mathfrak{M}$. One is the monoid presented in Example 4.4.1. We have seen that we can write any word in this monoid in the form $b^p a^q c^r$.

The other three generator element of $\mathfrak{M}$ is the monoid

$$\langle a,b,c: abc = ba,\ ac = ca,\ bc = cb \rangle$$

Any word in this monoid can be rewritten in the form $a^p b^q c^r$.

With the ability to write any word in either of the monoids in the forms given we can now apply exactly the same methods as in 4.3. Although, in both cases, we are 'missing' a relation the 'missing' relation is never required.

We now consider an example in a six generator element of M. We will see that in this case we cannot get round the problem by changing the order in which the words are rewritten. Intuitively, there are too many 'missing' relations.

Example 4.4.3:

Suppose $P \in \mathfrak{M}$ is defined as follows:

$$\begin{aligned} P = \langle a,b,c,d,e,f: & abd = ba,\ ace = ca,\ bcf = cb, \\ & dx = xd\ (x \in \{a,b,c,d,e,f\}), \\ & ex = xe\ (x \in \{a,b,c,d,e,f\}), \\ & fx = xf\ (x \in \{a,b,c,d,e,f\}) \rangle \end{aligned}$$

Now suppose that we are given any word of P then using the relations of P we can rewrite it in the form $a^p b^q c^r d^s e^t f^u$. For

example if w = abcabcabc then we can rewrite it as $w = a^3b^3c^3d^3e^3f^3$.

However, let us now consider the monoid Q ∈ $\mathfrak{M}$ constructed by replacing the relation 'ace = ca' in P by the relation 'ac = cae'. In Q the word abcabcabc cannot be rewritten in the form $a^Pb^Qc^Rd^Se^Tf^U$ because we cannot move the a's past the c's.

It is easy to see that in this case, although we cannot write a word in the form $a^Pb^Qc^Rd^Se^Tf^U$, we can write any given word in the form $a^{P_1}b^{Q_1}c^{R_1}\ a^{P_2}b^{Q_2}c^{R_2}\ .\ .\ .\ a^{P_n}b^{Q_n}c^{R_n}\ d^Se^Tf^U$. This is because if we attempt to employ the method used in Example 4.4.1 (of changing the order in which the words are rewritten) we readily see that a must go to the left of b, b must go to the left of c and c must go to the left of a. i.e. we have to write any word in terms of $a^{P_i}b^{Q_i}c^{R_i}$ as above (or a cyclic alternative such as $b^{Q_i}c^{R_i}a^{P_i}$, $c^{R_i}a^{P_i}b^{Q_i}$).

This means that given any equation of Q the development of a method to solve that equation must involve the substitution of words of the above form, $a^{P_1}b^{Q_1}c^{R_1}\ a^{P_2}b^{Q_2}c^{R_2}\ .\ .\ .\ a^{P_n}b^{Q_n}c^{R_n}d^Se^Tf^U$, for each variable. This suggests that we would have to substitute $a^Pb^Qc^Rd^Se^Tf^U$ for each variable and check for a solution, then we would have to substitute $a^{P_1}b^{Q_1}c^{R_1}\ a^{P_2}b^{Q_2}c^{R_2}\ d^Se^Tf^U$ and so on. If a solution exists we may be able to find it but the format of the substitutions strongly suggests that in general the problem will be undecidable.

The corresponding monoid in $\mathfrak{N}$ is

N = <a,b,c,d,e,f,d',e',f': abd = ba, ace = ca, bcf = cb,
abd' = ba, ace' = ca, bcf' = cb,

$$dx = xd \ (x \in \{a,b,c,d,e,f\}),$$

$$ex = xe \ (x \in \{a,b,c,d,e,f\}),$$

$$fx = xf \ (x \in \{a,b,c,d,e,f\}) >$$

We can see that the problem does not occur in this structure by rewriting abcabcabc as $a^3b^3c^3d^3e^3f^3$.

We now briefly discuss the extension of the work presented in this chapter in the same directions that we extended the work for nilpotent groups in Chapter 2.

4.5 Extensions of the above work

4.5.1 Equation solving in partially commutative monoids of higher class

In 4.3 we presented an algorithm for solving equations in the members of $\mathcal{N}$ relative to having an algorithm to solve ρ-systems. Any $N \in \mathcal{N}$ is a three sorted structure (any generator belongs to the set A,B or B'). These correspond to the "idea" of a nilpotent monoid of class 2. Although as we have indicated the word "nilpotent" involves the use of quotient structures which we are not allowed to use when considering monoids.

It is a straightforward task to consruct similar many sorted structures which would correspond to the "idea" of higher nilpotency class. The construction of equation solving algorithms in such structures has been left as an open research problem. This is because of time constraints on the research project rather than the difficulty of the problem.

4.5.2 Addition of the axiom $x^p = 1$

In Chapter 2 we discussed special p-groups as studied by U.Martin [47]. We saw that the inclusion of the axiom $x^p = 1$ to the theory of nilpotent groups of class 2 made the search space of the τ-equations finite. Thus we had an outright algorithm rather than a relative one.

If we attempt to use the same idea here and include the relation $x^p = 1$ $(x \in A \cup B \cup B')$ to the elements of $\mathcal{N}$ to create a new class of structures, $\mathcal{P}$ say, then we see that the elements of $\mathcal{P}$ are actually groups and as such have been covered in Chapter 2. The inverse of any element x is x^{p-1}.

4.5.3 Problems in just one variable

Repin [57] studied algorithmic equation solving for problems in only one variable for nilpotent groups of class 2. It was seen in Chapter 2 that restriction to this class of problems resulted in only linear diophantine equations occurring. As there is an algorithm to solve systems of linear diophantine equations this means that there exists an algorithm to solve this restricted class of problems. The question of whether or not the same thing happens when restricting to problems in just one variable in the members of $\mathcal{N}$ arises naturally. The answer to this question is that the above algorithm is an outright algorithm for this class of problems. The reason is because the linear diophantine equations that occur as part of a ρ-system are in one variable only thus giving a constant solution (if one exists). This can be substituted into the ρ-equations of the ρ-system to give a solution. We will illustrate this by considering the following example:

Example 4.5.3.1:

Suppose we have the same monoid that we considered in Example 4.3.7: $L = \langle a,b,c,c' : ac = ca,\ bc = cb,\ ac' = c'a,\ bc' = c'b,\ ab = bac,\ abc' = ba \rangle \in \mathcal{N}$.

Now consider the problem $xbx = ba^4$? in L.

If we let $x = a^{x_a}b^{x_b}c^{x_c}c'^{x_{c'}}$ (where $x_a, x_b, x_c, x_{c'}$ are non-negative integer variables) we obtain

$$a^{x_a}\, b^{x_b+1}\, a^{x_a}\, b^{x_b}\, c^{2x_c}\, c'^{2x_{c'}} = ba^4$$

Hence we have

$$a^{2x_a}\, b^{2x_b+1}\, c^{2x_c-x_a(x_b+1)}\, c'^{2x_{c'}} = ba^4$$

$$\text{or } a^{2x_a} b^{2x_b+1} c^{2x_c} c'^{2x_{c'}+x_a(x_b+1)} = ba^4$$

This means that we can derive one of the following two ρ-systems:

(1) $\{2x_a = 4,\ 2x_b + b = b,\ 2x_c - (x_b+1)x_a = 0,\ 2x_{c'} = 0\}$

(2) $\{2x_a = 4,\ 2x_b + b = b,\ 2x_c = 0,\ 2x_{c'} + (x_b+1)x_a = 0\}$

For (1) we have $x_a = 2$, $x_b = 0$, $x_c = 1$, and $x_{c'} = 0$.

For (2) we have $x_a = 2$, $x_b = 0$, $x_c = 0$, and $x_{c'} = -1$ (not allowed).

∴ the solution to the equation is $x = a^2c$

4.5.4 The cardinality type of any member of N

As can be seen in 4.3 the solution of equations in any member of N is dependent upon solving ρ-systems. Thus the cardinality type of the members of N is dependent entirely upon the cardinality type of ρ-systems. In Example 4.3.7 we saw that the ρ-system that occurred was in fact a system of linear diophantine equations. Also in Example 4.5.3.1 we derived a system of linear diophantine equations. When this occurs there is a single most general solution. The following example shows the reduction of an equation in L (the monoid considered in examples 4.3.7 and 4.5.3.1) to a 'proper' ρ-system i.e. one containing non-linear ρ-equations.

Example 4.5.4.1:

Recall that L is the following member of N

$$< a,b,c,c' : ac = ca,\ bc = cb, ac' = c'a,\ bc' = c'b,$$
$$ab = bac,\ abc' = ba >.$$

Suppose we have the problem $axyzb = a^3b^5c^7c'^2$? in L.

Applying the usual variable substitutions we have

$$a^{x_a+1}b^{x_b}a^{y_a}b^{y_b}a^{z_a}b^{z_b+1}\ c^{x_c+y_c+z_c}\ c'^{x_{c'}+y_{c'}+z_{c'}} = a^3b^5c^7c'^2$$

Using the algorithm we obtain

$$a^{x_a+y_a+z_a+1}\ b^{x_b+y_b+z_b+1}\ c^{x_c+y_c+z_c}\ c'^{x_{c'}+y_{c'}+z_{c'}+y_ax_b+z_a(x_b+y_b)}$$
$$= a^3b^5c^7c'^2$$

This provides us with the following non-linear ρ-system

$$x_a+y_a+z_a = 2,\ \ x_b+y_b+z_b = 4,\ \ x_c+y_c+z_c = 2,$$

$$x_{c'}+y_{c'}+z_{c'}+y_ax_b+z_a(x_b+y_b) \ = 2.$$

In the case of groups the troublesome diophantine equations only occurred under certain conditions (i.e. when the word to be unified with 1 lies in the commutator subgroup). The conditions, in members of $\mathcal{N}$, when the troublesome ρ-equations occur have not been specifically characterised in this way. This has been left as an open problem. Again the reason is time constraints on the project rather than the difficulty of the problem.

The next chapter concludes the thesis by providing a summary of the results presented and by indicating some open research problems that have arisen from this research project.

Chapter 5: Conclusion

5.1 Summary of Results

The main aim of this thesis was to study unification and equation solving with respect to partially commutative theories and structures. As we have seen, this partial commutativity was provided by the concept of nilpotency in the case of groups. The group theoretic problems have been reduced to number theoretic problems. The number theory has been left as an open research problem. We will now consider each chapter of the thesis in turn and highlight the main results that have arisen out of this research project.

5.1.1 Chapter 1

It has been indicated earlier that Chapter 1 is an introductory survey of the area. Having said this Theorem 1.4.2, although a known result, is not stated and proved explicitly in the unification literature. Indeed in some of the literature there appears to be some confusion over unification in a theory and equation solving in a model of the theory. For this reason the proof has been included.

5.1.2 Chapter 2

A unification algorithm was constructed for the theory of nilpotent groups of class k by constructing an equation solving algorithm for the free nilpotent group of class k. This algorithm depends upon the existence of an algorithm to solve certain kinds of diophantine equations of degree k. The main points of this chapter are:

5.1.2.1: If we have an algorithm to solve nth power free diophantine equations of degree n for $1 \leq n \leq k$ then we have an algorithm to find all solutions to any unification problem in the theory of nilpotent groups of class k.

5.1.2.2: N_2G is nullary.

5.1.2.3: In general the cardinality type of MN_kG is dependent upon the cardinality types of nth power free diophantine equations of degree n for $2 \leq n \leq k$. It is in any case infinitary.

5.1.3 Chapter 3

In this chapter we concerned ourselves with the question of the decidability of the unification problem in nilpotent groups. The main result of this chapter is that this problem is undecidable for the theory of nilpotent groups of class k where $k \geq 5$.

5.1.4 Chapter 4

In this chapter we turned our attention to partially commutative monoids and in particular to the class $\mathcal{N}$. The main result of this chapter is:

If we have an algorithm to solve ρ-systems then we have an algorithm to solve any equation in any member of $\mathcal{N}$.

5.2 Open Research Problems

There are many avenues of research that have been opened up by the work presented in this thesis. Some of these have been outlined already. However, we will collect all the most obvious open problems together in this section whether or not they have already been mentioned. Some of these problems appear reasonably straightforward and, as has also been indicated, have not been tackled because of time constraints on the research project.

This research project has given, in Chapters 1 and 4, various equation solving algorithms which are relative to having algorithms that can solve particular systems of diophantine equations. An obvious open problem is to solve the number theoretic problems that arise from this research. By the results of Chapter 3 we are only interested in solving the nth power free diophantine equations of degree n that occur in Chapter 2 for $2 \leq n \leq 4$. Another problem is to characterise the systems of equations occurring for class 3 and 4 in the way that we characterised the τ-equations for class 2.

The decidability of the unification problem for nilpotent groups of class 2, 3 and 4 depends upon the decidability of the corresponding systems of diopohantine equations. Similarly the decidability of the equation solving problem for members of $\mathcal{N}$ depends upon the decidability of ρ-systems. If such algorithms could be constructed then the relative algorithms presented in this thesis would become actual ones. Another possible research problem would be to consider the complexity of the algorithms presented in this thesis.

We saw in Chapter 2 that we have an outright unification algorithm for one-variable problems in a free nilpotent group of class 2 (Repin [57]). This is not the case for free nilpotent

groups of class 3. Here we are dependent upon certain diophantine equations of degree 2. A possible direction of research is to investigate further the type of equations that occur here (in the way that we developed the τ-equations for the general case in class 2). It is also feasible to investigate one-variable problems in nilpotent groups of higher class. In Chapter 3 we considered another result established by Repin [57] which says that the problem of solving any equation, involving only one variable, in a free nilpotent group of class k for $k \geq 10^{20}$ is undecidable. It is extremely likely that the value of k in this result can be reduced. It may be possible to do this by employing some of the methods developed in this thesis.

In Chapter 4 we considered equation solving in members of $\mathcal{N}$ i.e. certain partially commutative monoids. These correspond to nilpotent structures of class 2. A natural extension of this work would be to consider equation solving in similar partially commutative monoids corresponding to structures of higher nilpotency class. Also it would be interesting to investigate the conditions, in members of $\mathcal{N}$, when the troublesome ρ-equations occur.

References

[1] H.Abdulrab and J.P.Pecuchet, Solving Word Equations, Journal of Symbolic Computation 1990.

[2] H.Ait-Kaci and M.Nivat (editors), Resolution of Equations in Algebraic Structures, Academic Press 1989.

[3] F.Baader, Unification in Idempotent Semigroups is of Type Zero, Journal of Automated Reasoning Vol. 2 (1986), pages 283-286.

[4] F.Baader, Unification in Commutative Theories, Journal of Symbolic Computation 1989.

[5] G.Birkhoff, On the Structure of Abstract Algebras, Proceedings of the Cambridge Phil. Soc. Vol. 31 (1935), pages 433-454.

[6] H-J.Buckert, A.Herold, D.Kapur, J.Sielmann, M.Stickel, M.Tepp and H.Zhang, Opening the AC-Unification Race, Journal of Automated Reasoning Vol. 4 (1988), pages 465-474.

[7] H-J.Buckert, A.Herold and M.Schmidt-Schauss, On Equational Theories, Unification and Decidability, Proceedings of the 2nd International Conference on Term Rewriting Techniques and Applications, Springer L.N.C.S. Vol.256 (1987), pages 204-215.

[8] W.Buttner, Unification in Finite Algebras is Unitary, Proceedings of C.A.D.E. 9, Springer-Verlag 1987, pages 205-368.

[9] J.Cannon, An Introduction to the Group Theory, Language Cayley, in Computational Group Theory (edited by M.D.Atkinson), Academic Press 1984, pages 145-183.

[10] C.Chang and R.C.Lee, Symbolic Logic and Mechanical Theorem Proving, Academic Press 1973.

[11] J.Cohen, Constraint Logic Programming Languages, Communications of the A.C.M. Vol.33 No. 7 (1990), pages 52-68.

[12] A.G.Cohn, A More Expressive Formulation of Many Sorted Logic, Journal of Automated Reasoning Vol.3 (1987), pages 113-200.

[13] A.Colmerauer, A Prolog II Reference Manual and Theoretical Model, Rep. Group d'Intelligence Artificielle (1982), Universite d'Aix-Marseille II, Luminy.

[14] A.Colmerauer, An Introduction to Prolog III, Communications of the A.C.M. Vol.33 No. 7 (1990), pages 69-90.

[15] D.DeGroot and G.Lindstrom (editors), Logic Programming: Functions Relations and Equations, Prentice-Hall 1986.

[16] M.Dincbas et. al., The Constraint Logic Programming Language CHIP, Proceedings of the International Conference on 5th Generation Computer Systems, Ohmsha (1988), pages 693-702.

[17] F.Fages and G.P.Huet, Complete Sets of Unifiers and Matchers in Equational Theories, Proceedings of the Colloquium on Trees in Algebra and Programming, Springer L.N.C.S. Vol. 159 (1983).

[18] R.Feldman and M.C.Golumbic, Optimization Algorithms for Student Scheduling via Constraint Satisfiability, The Computer Journal Vol.33 No. 4 (1990), pages 356-364.

[19] J.Goguen and J.Meseguer, EQLOG: Equality, Types and Generic Modules for Logic Programming, in [15] pages 295-363.

[20] W.E.Gould, A Matching Procedure for ω-Sorted Logic, Scientific Report no.4, Airforce Cambridge Research Laboratories, Bedford, Massachusetts (1966).

[21] M. Hall, The Theory of Groups, Macmillan 1964.

[22] P. Hall, Some Word Problems, J. London Math. Soc. 33 (1958), pages 482-496.

[23] A.G.Hamilton, Logic for Mathematicians, Cambridge University Press 1988.

[24] A.Herold, Combination of Unification Algorithms, Proceedings of the 8th Conference on Automated Deduction 1986, Springer L.N.C.S. Vol.230, pages 450-469.

[25] A.Herold and J.Siekmann, Unification in Abelian Semigroups, Journal of Automated Reasoning Vol. 3 (1987), pages 247-283.

[26] J.-M. Hullot, Canonical Forms and Unification, Proceedings of the 5th Conference on Automated Deduction, Springer L.N.C.S. Vol. 87 (1980), pages 318-334.

[27] S.Holldobler, Foundations of Equational Logic Programming, L.N.A.I. Vol. 353 Springer-Verlag 1989.

[28] G.Huet, An Algorithm to Generate the Basis of Solutions to Homogeneous Linear Diophantine Equations, Information Processing Letters Vol. 7 (1978), pages 144-147.

[29] J.Jaffar, Minimal and Complete Word Unification, Journal of the Association for Computing Machinary 1990.

[30] J.Jaffar and J-L.Lassez, Constraint Logic Programming, Proceedings of the 14th POPL Conference 1987, pages 111-119.

[31] J.P.Jones, Universal Diophantine Equations, Journal of Symbolic Logic Vol. 47 (1982), pages 549-571.

[32] J.-P.Jouannaud and C.Kirchner, Solving Equations in Abstract Algebras: A Rule-Based Survey of Unification, preprint.

[33] C.Kirchner, From Unification in a Combination of Equational Theories to a New AC-Unification Algorithm, in [2], pages 171-210.

[34] C.Kirchner, H.Kirchner and M.Rusinowitch, Deduction with Symbolic Constraints, preprint.

[35] D.E.Knuth, The Art of Computer Programming Vol.2 (2nd edition), Addison-Wesley 1981.

[36] D.E.Knuth and P.B.Bendix, Simple Word Problems in Universal Algebras, in [40], pages 263-298.

[37] W.A.Kornfeld, Equality for Prolog, in [15], pages 279-293.

[38] E.F.Krause, On the Collection Process, Proceedings of the American Mathematical Society Vol.15 (1964), pages 497-504.

[39] D.S. Lankford, G. Butler and B. Brady; Abelian Group Unification Algorithms for Elementary Terms, Contemporary Mathematics Vol. 29 1984.

[40] J.Leech (editor), Computational Problems in Abstract Algebra, Pergamon Press 1970.

[41] A.Lentin, Equations dans le Monoide libre, Gauthier-Villars 1972.

[42] J.W.Lloyd, The Foundations of Logic Programming, 2nd edition, Springer-Verlag 1987.

[43] G.S. Makanin, The Problem of Solvability of Equations in a Free Semigroup, Akad. Nauk. SSSR Vol. 233 (1977).

[44] G.S.Makanin, Equations in a Free Group, Math. USSR Izvestiya Vol. 21 No. 3 (1983), 483-546.

[45] Y.Malachi, Z.Manna and R.Waldinger, TABLOG: A New Approach to Logic Programming, in [15], pages 365-394.

[46] A.Martelli and U.Montanari, An Efficient Unification Algorithm, ACM Transactions on Programming Languages and Systems Vol. 4 (1982), pages 258-282.

[47] U.Martin, Unification in Special p-groups, preprint 1990.

[48] U.Martin and T.Nipkow, Unification in Boolean Rings, Proceedings of the 8th International Conference on Automated Deduction, Springer LNCS Vol.230 (1986), pages 506-513.

[49] U.Martin and T.Nipkow, Boolean Unification - The Story So Far, Journal of Symbolic Computation Vol.7 (1989), pages 275-294.

[50] Yu. V. Matiyasevitch, Enumerable Sets are Diophantine, Dokl. Akad. Nauk SSSR 191, No. 2 (1970), pages 279-282.

[51] E.Mendholson, Introduction to Mathematical Logic, D.Van Nostrand Company 1966.

[52] J.Neubuser, Investigations of Groups on Computers, in [40], pages 1-20.

[53] H. Neumann, Varieties of Groups, Springer-Verlag 1967.

[54] U.Nilsson and J.Matuszynski, Logic, Programming and Prolog, Wiley 1990.

[55] G.Plotkin, Building in Equational Theories, Machine Intelligence Vol.7 (1972), pages 73-90.

[56] P.Raulefs and J.Siekmann, Unification of Idempotent Functions, Internal Report MEMO SEKI-78-I, Universitat Karlsruhe (1978).

[57] N.N.Repin, The Solvability Problem for Equations in One Unknown in Nilpotent Groups, Math. USSR IZV. 25 (1985), pages 601-618.

[58] J.A.Robinson, A Machine Orientated Logic Based on the Resolution Principle, Journal of the Association for Computing Machinary Vol.12 (1965), pages 23-41.

[59] J.A.Robinson and L.Wos, Paramodulation and Theorem Proving in First Order Theories with Equality, Machine Intelligence Vol.4 (1969), pages 135-150.

[60] V.A.Romankov, Unsolvability of the Endomorphic Reducibility Problem in Free Nilpotent Groups and in Free Rings, Algebra and Logic Vol.16 (1977), pages 310-320.

[61] V.A. Romankov, Equations in Free Metabelian Groups, Siberian J. Math. Vol 20 Part 1 (1979), pages 469-471.

[62] M.Schmidt-Schauss, Unification Under Associativity and Idempotence is of Type Nullary, Journal of Automated Reasoning Vol. 2 (1986), pages 277-282.

[63] M.Schmidt-Schauss, Unification in Many Sorted Equational Theories, Proceedings of the 8th International Conference on Automated Deduction, Springer LNCS Vol.230 (1986), pages 538-552.

[64] M.Schmidt-Schauss, Combination of Unification Algorithms for Equational Theories with Free Function Symbols, SEKI-Report, University of Kaiserslautern (1987).

[65] J.H. Siekmann; Unification of Commutative Terms, Proceedings of the International Symposium on Symbolic and Algebraic Manipulation 1979, Springer LNCS Vol. 72, pages 531-545.

[66] J.H. Siekmann; Unification Theory, Proceedings of the 8th European Conference on Artificial Intelligence Vol.2 (1986), pages 6-25.

[67] J.H. Siekmann; Unification Theory, Journal of Symbolic Computation Vol. 7 (1989), pages 207-274 (this is an updated version of [66]).

[68] M.E.Stickel, A Complete Unification Algorithm for Associative Commutative Functions, Proceedings of the 4th International Joint Conference on Artificial Intelligence (1975), pages 71-82.

[69] M.E.Stickel, A Unification Algorithm for Associative Commutative Functions, Journal of the Association for Computing Machinary Vol. 28 (1981), pages 423-434.

[70] P.Szabo, Theory of First Order Unification, Ph.D. Thesis (1982), Universitat Karlsruhe.

[71] P.Szabo and E.Unvericht, The Unification Problem for Distributive Terms, Internal Report SEKI-78-05, Institut fur Informatik I, Universitat Karlsruhe.

[72] C.Walther, A Many-Sorted Calculus Based on Resolution and Paramodulation, Research Notes in Artificial Intelligence, Kauffman 1983.

[73] K.Yellick, Combining Unification Algorithms for Confined Regular Theories, Proceedings of the 1st International Conference on Rewriting Techniques and Applications, Springer L.N.C.S. Vol.202 (1985), pages 365-380.