Formal Languages and the Word Problem for Groups

Iain A. Stewart[*] and Richard M. Thomas[†]
Department of Mathematics and Computer Science,
University of Leicester, Leicester LE1 7RH, England.

1 Introduction

The aim of this article is to survey some connections between formal language theory and group theory with particular emphasis on the word problem for groups and the consequence on the algebraic structure of a group of its word problem belonging to a certain class of formal languages. We define our terms in Section 2 and then consider the structure of groups whose word problem is regular or context-free in Section 3. In Section 4 we look at groups whose word-problem is a one-counter language, and we move up the Chomsky hierarchy to briefly consider what happens above context-free in Section 5. In Section 6, we see what happens if we consider languages lying in certain complexity classes, and we summarize the situation in Section 7. For general background material on group theory we refer the reader to [25, 26], and for formal language theory to [7, 16, 20].

2 Word problems and decidability

In this section we set up the basic notation we shall be using and introduce the notions of "word problems" and "decidability".

In order to consider the word problem of a group as a formal language, we need to introduce some terminology. Throughout, if Σ is a finite set (normally referred to in this context as an *alphabet*) then we let Σ^* denote the set of all finite words of symbols from Σ, including the *empty word* ϵ, and Σ^+ denote the set of all non-empty finite words of symbols from Σ. To put this another way, Σ^* is the free monoid generated by Σ, and Σ^+ is the free semigroup generated by Σ. A subset of Σ^* is known as a *language*.

One of the central notions in combinatorial group theory is that of the *word problem* of a group G. Given a (finite) generating set X for G, every word over the alphabet $\Sigma = X \cup X^{-1}$ represents a unique element of G, and we have a natural homomorphism from the free monoid Σ^* onto G. The word problem

[*]e-mail: i.a.stewart@mcs.leicester.ac.uk
[†]e-mail: rmt@mcs.leicester.ac.uk

$W_X(G)$ of G (with respect to the generating set X) is defined to be the set of all words of Σ^* that represent the identity element of G. This is a slightly different approach to the more usual one which is to think of the word problem as being the question as to whether or not a word α represents the identity: by defining the word problem to be a set of words, we transfer the question to deciding membership of this set, and this approach fits more naturally in the context of formal language theory.

A classic question was that of the *decidability* of the word problem: given a finitely presented group G, is there an algorithm that decides whether or not a given word $\alpha \in \Sigma^*$ belongs to $W_X(G)$? In order to make sense of this, we should briefly explain what we mean by "decidable".

There are many (equivalent) models of computation, but we shall take the Turing machine here since it is in the spirit of the other models we will be considering. Further, there are many versions of a Turing machine, and we choose one that will suit our needs. Our Turing machines have a read-only input tape which, for an input of length n, consists of $n+2$ tape cells, the first of which contains the "left-end-of-tape" marker $\triangleright$, the last of which contains the "right-end-of-tape" marker $\triangleleft$ and such that the cells between hold the input word. They also have a fixed number of work tapes on which we perform the computation. The work tapes initially contain $\triangleright$ in their leftmost cells, are blank everywhere else and are unbounded in length to the right. We may move freely over these work tapes and read from and write to the cells as we wish. Initially, the work heads are positioned over the leftmost cells of the work tapes (i.e., those initially holding the symbol $\triangleright$) and the input head is positioned over the cell adjacent to the leftmost cell of the input tape (i.e., over the cell holding the first symbol of the input string, if the input string is non-empty, or over the cell holding the symbol $\triangleright$ otherwise). More formally, a *deterministic Turing machine* (DTM) M is a tuple $(Q, \Sigma, \Gamma, \delta, s, h)$ where Q is a finite set of "states", Σ is a finite set of input symbols, and Γ is a finite set of work tape symbols, including $\triangleright$, $\triangleleft$ and Δ (where Δ is the blank symbol). In addition, s and h are designated states (the "start state" and the "halt state") and δ is a partial function from

$$Q \times \Sigma \times \Gamma^m \to Q \times \{L, R, N\} \times (\Gamma \times \{L, R, N\})^m$$

so that if $\delta(q, x, g_1, \dots, g_m) = (r, d, g'_1, d_1, \dots, g'_m, d_m)$, where m is the number of work tapes, we imagine that, when M is in state q reading a symbol x on the input tape and $g_1, \dots, g_m$ on the respective work tapes, then: M erases $g_1, \dots, g_m$ and writes $g'_1, \dots, g'_m$ in their place; changes state to r; moves that input head in the direction indicated by d (left if $d = L$, right if $d = R$ and not at all if $d = N$); and moves the input heads in the directions indicated by $d_1, \dots, d_m$. If ever M attempts to move left on any tape when reading the leftmost cell then m crashes as it does if ever it attempts to move the input head right when reading the rightmost cell of the input tape, and there is no transition defined from the halt state. If M is set up in the start state s with input word α then α is said to be *accepted* if M reaches the halt state h and *rejected* otherwise (i.e., if the machine either hangs, crashes or else runs indefinitely without entering the halt state). The *language* $L(M)$ of M is the set of all words accepted by M: we say that a language L is *semi-decidable* or *recursively enumerable* if $L = L(M)$ for some DTM M.

One way of looking at this is to imagine that the input word α represents a question to which we want an answer, either "yes" or "no", and that the DTM M accepts α if, and only if, the answer is "yes". However, with this notion, we may never know if the answer is "no" as a Turing machine may fail to halt on some input. We can modify our definition of a Turing machine so as to have two halt states h_y and h_n, and insist that such a machine halts if it enters either of these states. Furthermore, we can design such Turing machines so that for any input α, they necessarily reach one of these two states. We define the *yes-language* $Y(M)$ of such a Turing machine M to be the set of all input words such that M reaches h_y (and the *no-language* $N(M) = \Sigma^* \setminus Y(M)$ of M to be the set of all input words such that M reaches h_n) and call such a machine a *decision-making DTM*. We say that a language L is *decidable* or *recursive* if $L = L(M)$ for some decision-making DTM M. Two standard results in computability theory state that a recursive language is necessarily recursively enumerable, but not conversely, and that given a (encoding of a) description of a Turing machine as input, there does not exist a decision-making Turing machine which answers "yes" if the input machine is a decision-making Turing machine and "no" otherwise.

One should comment here, in passing, that the computational power of our Turing machines is not increased if one introduces "non-determinism". We define a non-deterministic Turing machine (NTM) $M = (Q, \Sigma, \Gamma, \delta, s, h)$ in the same way as a DTM, except that, as opposed to a partial function $\delta : Q \times \Sigma \times \Gamma^m \to Q \times \Gamma \times \{L, R, N\} \times (\Gamma \times \{L, R, N\})^m$, we allow δ to be a subset of $Q \times \Sigma \times \Gamma^m \times Q \times \Gamma \times \{L, R, N\} \times (\Gamma \times \{L, R, N\})^m$. The idea here is that there may be choice at some instant as to which transition the machine performs, and we say that a word α is *accepted* if at least one possible computation path reaches the halt state h. This does not lead to an increase in power (in the sense of what can be computed), in that any language accepted by a NTM is also accepted by a DTM (and, similarly, any language which is the yes-language of a decision making NTM, where a word is accepted if some computation path leads to the halt state h_y, is the yes-language of a decision-making DTM).

It is not too difficult to see that in any finitely-presented group, the word problem is semi-decidable: one systematically enumerates all words representing the identity and compares each in turn with the input word, stopping if and when we get a match. However, a more interesting question is that of whether or not finitely presented groups necessarily have a decidable word problem, or, to put this question another way, is the set $W_X(G)$ necessarily a decidable language for a finitely presented group G (with generator set X)?

The answer to this question was shown to be "no" by Novikov [30] and Boone [8] (independently) in the 1950's, and much subsequent research tended to focus on whether or not the word problem is decidable if we impose restrictions on our class of groups (e.g., to the class of automatic groups, where the word problem is decidable [13], or to the class of solvable groups of derived length three, where the word problem is undecidable [22]), and also the degrees of undecidability of the word problem for various groups or classes of groups, along with an analogous analysis of related problems like the conjugacy problem, the generalized word problem and the isomorphism problem (see [27] for an elegant survey of such results).

We will be looking at degrees of decidability, focussing on what happens when the word problem can be decided by a computing device less powerful than a Turing machine. We will be particularly interested in the consequences on the algebraic structure of such a group. As far as groups with a decidable word problem are concerned, we have the following fascinating result of Boone and Higman [9]:

THEOREM 2.1 *Let G be a finitely generated group. Then the word problem of G is decidable if, and only if, there exists a simple group H and a finitely presented group K with $G \leq H \leq K$.*

This was generalized in [37]:

THEOREM 2.2 *Let G be a finitely generated group. Then the word problem of G is decidable if, and only if, there exists a finitely generated simple group H and a finitely presented group K with $G \leq H \leq K$.*

We will survey some similar results when we start restricting the range of languages for the word problem.

3 Recognizable and context-free word problems

In the previous section we discussed the idea of decidability with respect to word problems. However, an alternative approach to classifying word problems was suggested by work of Anisimov [1] who classified groups with a regular word problem. Essentially a *regular language* is one accepted by a *finite state automaton*, i.e., a Turing machine that has no work tapes and where the head on the input tape always moves right on every move. It is conventional here to replace the idea of a halt state with a set F of *accept states*: we continue reading the input α, accepting α if we end up in a state in F after reading the last symbol in α and rejecting α otherwise. As with Turing machines, allowing non-determinism does not increase the range of languages accepted. We let $\mathcal{R}$ denote the class of all regular languages. It turns out that no matter which generating set X we choose for G, either $W_X(G) \in \mathcal{R}$ for all such generating sets or for none. We can now state the result from [1]:

THEOREM 3.1 *Let G be a finitely generated group. Then $W(G)$ is regular if, and only if, G is finite.*

In this way, the focus was shifted from degrees of undecidability to degrees of decidability. Anisimov's result implied a more general research plan of linking the structure of a group with the nature of its word problem as a formal language. To this end, another significant related result was obtained by Muller and Schupp [28] who classified (if we blend in a deep result of Dunwoody [12]) groups with a context-free word problem. In order to explain the term "context-free", we need some more definitions.

We can extend our finite state automaton by adding a *stack*. We now assume that our finite state automaton has a single work tape where access to the symbols of this work tape is restricted. The idea is that we may only read the rightmost non-blank cell of the work tape at any point in time. We may delete the contents of this cell and move the work head left, if we are not scanning the leftmost cell of the work tape, i.e., "pop an element off the stack", or we may move the work head one cell to the right and write a new symbol into this cell, i.e., "push an element onto the stack". In this way the contents of the stack are always of the form $\rhd\gamma$ for some (possibly empty) string γ with the head positioned over the rightmost symbol in $\rhd\gamma$. An extension of a non-deterministic finite state automaton in this way is known as a *pushdown automaton*. Unlike Turing machines and finite state automata, insisting on determinism *does* restrict the range of languages accepted (see [20] for example), and we have the class $\mathcal{CF}$ of *context-free languages* accepted by pushdown automata and the class $\mathcal{DCF}$ of *deterministic context-free languages* accepted by deterministic pushdown automata. It is clear that $\mathcal{R} \subseteq \mathcal{DCF}$, and, in fact, $\mathcal{R} \subset \mathcal{DCF}$.

The fact that $\mathcal{R} \subset \mathcal{DCF}$ is a standard result in formal language theory, and is reflected here by the fact that recognizing word problems by pushdown automata allows a wider class of groups than the finite groups. We say that a finitely-generated group G is *context-free* if $W(G) \in \mathcal{CF}$ and *deterministic context-free* if $W(G) \in \mathcal{DCF}$ (as for regular languages, it turns out that as to whether $W_X(G) \in \mathcal{CF}$ or $W_X(G) \in \mathcal{DCF}$ is independent of the choice of generating set X). For instance, it is reasonably clear that a finitely generated free group is deterministic context-free: as one reads the input word, the generators are pushed on the stack, unless one reads a generator x with x^{-1} on the top of the stack, in which case we merely pop x^{-1} off the stack. One can extend this to show that any finitely generated virtually free group (i.e., a finitely generated group with a free subgroup of finite index) is deterministic context-free.

Let G be a finitely generated group and let Γ be the Cayley graph of G with respect to some finite generating set. Let $\Gamma^{(n)}$ denote the set of vertices in Γ at distance at most n from the identity element. Then the number of *ends* of G is defined to be

$$\lim_{n\to\infty} (\text{the number of infinite components of } \Gamma - \Gamma^{(n)})$$

(which may be infinite). This number is, in fact, independent of the generating set chosen. An *accessible series* for G is a series $G = G_0 \geq G_1 \geq G_2 \geq \ldots \geq G_n$ of subgroups such that each G_i has a decomposition as a non-trivial free product with amalgamation, or as an HNN extension, where one of the factors or the base is G_{i+1}, and the amalgamated or associated subgroups are finite. A group G is *accessible* if there is a finite upper bound on the length n of such a series.

In [28], it was shown that an infinite context-free group has more than one end, and then, using Stallings' classification of finitely generated groups with more than one end [36], that G is virtually free if, and only if, G is context-free and accessible. It was proved in [12] that all finitely presented groups are accessible. Since every context-free group is finitely presented [2] and deterministic context-free [29], we have the following:

Theorem 3.2 *If G is a finitely-generated group, then*:

$$G \text{ is context-free} \Leftrightarrow G \text{ is deterministic context-free} \Leftrightarrow G \text{ is virtually free.}$$

See also [23, 35]. It is interesting that, despite the fact that there are languages that are context-free but not deterministic context-free, there are no such languages that are the word problems of groups!

For the classes of languages encountered so far, we have stated that whether the word problem of some group is in the class is independent of the generating set chosen for the group. We now give a general condition for this to be so.

Let $\mathcal{F}$ be a class of languages. We say that $\mathcal{F}$ is *closed under homomorphism* if:

$$L \in \mathcal{F} \cap \Sigma^*, \phi : \Sigma^* \to \Omega^* \text{ a monoid homomorphism } \Rightarrow L\phi \in \mathcal{F},$$

and that $\mathcal{F}$ is *closed under inverse homomorphism* if:

$$L \in \mathcal{F} \cap \Omega^*, \phi : \Sigma^* \to \Omega^* \text{ a monoid homomorphism } \Rightarrow L\phi^{-1} \in \mathcal{F}.$$

We also say that $\mathcal{F}$ is *closed under intersection with regular languages* if:

$$L_1 \in \mathcal{F} \cap \Sigma^*, L_2 \in \mathcal{R} \cap \Sigma^* \Rightarrow L_1 \cap L_2 \in \mathcal{F}.$$

We have the following well-known result (see [19], for example):

Proposition 3.3 *If $\mathcal{F}$ is a family of languages closed under inverse homomorphism, S and T are finite subsets of the group G, and $G = \langle S \rangle = \langle T \rangle$, then:*

$$W_S(G) \in \mathcal{F} \text{ if, and only if, } W_T(G) \in \mathcal{F}.$$

A *cone* is a family $\mathcal{F}$ of languages closed under homomorphism, inverse homomorphism and intersection with regular languages. It is well known that $\mathcal{R}$ and $\mathcal{CF}$ are both cones (see [20], for example). In particular, they are closed under inverse homomorphism, and so we can talk about the word problem of a group being "regular" or "context-free" without ambiguity. Whereas $\mathcal{DCF}$ is not a cone, it is closed under inverse homomorphisms.

4 One-counter groups

Returning to machines, a *one-counter automaton* is a pushdown automaton where there are only two stack symbols, $\triangleright$ and g say. At any stage, the stack of a one-counter automaton contains $\triangleright g^n$ for some $n \geq 0$, and so is effectively described by a single natural number n; hence the title "one-counter". If, in addition, the pushdown automaton is deterministic then we say that we have a *deterministic one-counter automaton*. We say that L is a *one-counter language* if $L = L(M)$ for some one-counter automaton M, and a *deterministic one-counter language* if $L = L(N)$ for some deterministic one-counter automaton N.

We let $\mathcal{OC}$ be the class of one-counter languages and $\mathcal{DOC}$ be the class of deterministic one-counter languages. Whereas $\mathcal{OC}$ is a cone (see [7] for example), $\mathcal{DOC}$ is not but it is closed under inverse homomorphisms (see [17, 20], for example). We say that a finitely generated group is *one-counter* if its word problem lies in $\mathcal{OC}$ and *deterministic one-counter* if its word problem lies in $\mathcal{DOC}$. The family of one-counter languages is particularly significant when studying the word problem in groups, as the next result from [17] shows:

THEOREM 4.1 *Let $\mathcal{F}$ be a cone with $\mathcal{F} \subseteq \mathcal{CF}$, and let Υ be the class of all finitely-generated groups G such that $W(G) \in \mathcal{F}$. Then Υ is the class of recognizable groups, one-counter groups or context-free groups.*

As far as one-counter groups are concerned, we have a result analogous to Theorem 3.2 from [17]:

THEOREM 4.2 *If G is a finitely-generated group, then*:

$$G \text{ is one-counter} \Leftrightarrow G \text{ is deterministic one-counter} \Leftrightarrow G \text{ is virtually cyclic.}$$

This result completes the classification of classes of groups whose word problem lies in a cone $\mathcal{F}$ with $\mathcal{F} \subseteq \mathcal{CF}$. Theorem 4.1 shows that the class of one-counter groups is a very natural one when one considers characterizations of groups in terms of their word problems. In addition, the infinite one-counter groups are precisely the groups with two ends. Notice that, in Theorem 4.2, as in Theorem 3.2, we have that the class of languages accepted by the deterministic model of the machine is a proper subclass of the class of languages accepted by the non-deterministic model, but this is not preserved when we restrict ourselves to considering the word problems of groups. One is tempted to ask to what extent this is a general phenomenon.

We saw in Proposition 3.3 that we really only need to consider classes of languages $\mathcal{F}$ closed under inverse homomorphisms to remove the problem that the word problem for a group lying in $\mathcal{F}$ might depend on the particular generating set chosen. This leads to the natural question as to whether there are any interesting classes $\mathcal{F}$ of languages closed under inverse homomorphism and contained in $\mathcal{CF}$ such that the class of groups whose word problem lies in $\mathcal{F}$ is not the class of recognizable, one-counter or context-free groups.

5 Above context-free

The situation higher up the Chomsky Hierarchy, above the context-free languages, seems to be much less clear. In particular, while many examples of groups with a *context-sensitive word problem* (a word problem that can be recognised by a non-deterministic Turing machine which uses $O(n)$ cells on its single work tape for an input of length n), or even a *deterministic context-sensitive word problem* (where the Turing machine is necessarily deterministic), are known (see [18], for example), as yet we have no structural classification as to precisely which groups have a context-sensitive word problem. As pointed out in [38], this may be quite a challenging problem.

Our discussion of interactions between formal language theory and group theory brings us naturally to the study of automatic groups. These groups were defined by Thurston as a class of finitely presented groups: he noticed that a result of Cannon [10] on the combinatorial structure of discrete hyperbolic groups could be re-expressed in terms of finite automata. We will want to consider automata accepting pairs (α, β) of words with $\alpha, \beta \in A^*$. If $\alpha = a_1 a_2 \ldots a_n$ and $\beta = b_1 b_2 \ldots b_m$, this is accomplished by having an automaton with input alphabet $A \times A$ and reading pairs (a_1, b_1), (a_2, b_2), and so on. To deal with the case where $n \neq m$, we introduce a *padding symbol* \$. More formally, we define a mapping $\delta_A : A^* \times A^* \to A(2, \$)^*$, where $\$ \notin A$ and $A(2, \$) = ((A \cup \{\$\}) \times (A \cup \{\$\})) - \{(\$, \$)\}$, by:

$$(\alpha, \beta)\delta_A = \begin{cases} (a_1, b_1) \ldots (a_n, b_n) & \text{if } n = m \\ (a_1, b_1) \ldots (a_n, b_n)(\$, b_{n+1}) \ldots (\$, b_m) & \text{if } n < m \\ (a_1, b_1) \ldots (a_m, b_m)(a_{m+1}, \$) \ldots (a_n, \$) & \text{if } n > m. \end{cases}$$

Given this, if G is a group, A is a finite set, L is a regular subset of A^*, and $\phi : A^* \to G$ is a homomorphism with $L\phi = G$ then we say that (A, L) is an *automatic structure* for G if $\{(\alpha, \beta) : \alpha, \beta \in L, \alpha = \beta\}\delta_A$ is regular, and $\{(\alpha, \beta) : \alpha, \beta \in L, \alpha a = \beta\}\delta_A$ is regular for each $a \in A$. If a group G has an automatic structure (A, L), for some A and L, then we say that G is *automatic* (an equivalent definition can be expressed in terms of the Cayley graph of the group).

The class of automatic groups includes finite groups, free groups, free abelian groups, various small-cancellation groups, word-hyperbolic groups and co-compact discrete groups of isometries of negatively curved space. The study of automatic groups is a thriving research area and the book [13] provides a standard reference (see also [6]). A significant result in this area is the following theorem from [32]:

THEOREM 5.1 *A finitely generated subgroup of an automatic group has deterministic context-sensitive word problem.*

Since a direct product of finitely generated free groups is automatic, this means that any finitely generated subgroup of $F_2 \times \ldots \times F_2$ has a deterministic context-sensitive word problem, so that there are examples of groups with a deterministic context-sensitive word problem that are not finitely presented. It would be interesting to know if context-sensitive is the optimal result here, i.e., is there a proper subfamily $\mathcal{F}$ of the deterministic context-sensitive languages which is closed under inverse homomorphism and such that the word problem for finitely-generated subgroups of automatic groups lies in $\mathcal{F}$? It is even an open question as to whether or not the deterministic context-sensitive languages form a proper subclass of the context-sensitive languages.

6 Complexity

The advent of electronic computers led to a realization that "decidable" need not mean "feasibly decidable", and a theory, *computational complexity theory*, has been developed whose aim is the precise classification of the degrees of decidability of problems, with particular emphasis on the resources needed. One motivation for studying groups whose word problem lies in a particular class of formal languages is that computation in the group might actually be practically, rather than just theoretically, possible. As can be seen above, there is a close relationship between classes of formal languages and the languages accepted by resource-bounded models of computation (for a general account of complexity theory, see [20, 31]).

Roughly, complexity theory has to it two aspects: the study of upper bounds and the study of lower bounds. The former are essentially concrete algorithms that have specific bounds on the resources used, and the latter are proofs that some specific problem can not be solved by some model of computation within some resource bounds. Lower bound results are notoriously difficult to obtain and often we make do with showing some problem to be "complete" for some "complexity class", as we now explain.

As in the rest of this paper, let us restrict ourselves here to the question of membership of a language L. We say that this problem can be solved in *deterministic time* $f(n)$ if there is a decision-making DTM M such that, given any input of length n, M terminates in at most $f(n)$ steps: we similarly say that the problem can be solved in *deterministic space* $g(n)$ if there is a decision-making DTM M such that, given any input of length n, M terminates using at most $g(n)$ cells on the work tape. We have similar definitions for *non-deterministic time* and *non-deterministic space*. Bounding resources in this way yields "complexity classes" such as: DTIME(*poly*), the class of languages L for which the membership problem is solvable in deterministic time $f(n)$, for some polynomial $f(n)$; NSPACE(*log*), the class of languages L for which the membership problem is solvable in non-deterministic space $O(\log n)$; and so on (recall, the complexity classes NSPACE($O(n)$) and DSPACE($O(n)$) are the classes of context-sensitive languages and deterministic context-sensitive languages, respectively). Complexity classes such as DTIME(*poly*) and NSPACE(*log*) are very robust in the sense that the same complexity class arises by restricting different (reasonable) models of computation in the same way.

At the fulcrum of complexity theory is the open question as to whether DTIME(*poly*) and NTIME(*poly*) are identical (also known as the P = NP question), although there are a great number of similar open questions concerning other complexity classes. For example, the basic hierarchy of complexity theory is:

$$\begin{aligned}\mathrm{DSPACE}(\mathit{log}) &\subseteq \mathrm{NSPACE}(\mathit{log}) \subseteq \mathrm{DTIME}(\mathit{poly}) \\ &\subseteq \mathrm{NTIME}(\mathit{poly}) \subseteq \mathrm{DSPACE}(\mathit{poly}) \subseteq \mathrm{NSPACE}(\mathit{poly}),\end{aligned}$$

and the only additional information known about these containments is

$$\mathrm{NSPACE}(\mathit{log}) \neq \mathrm{DSPACE}(\mathit{poly}) \text{ and } \mathrm{DSPACE}(\mathit{poly}) = \mathrm{NSPACE}(\mathit{poly}).$$

The questions as to whether the above containments are proper are generally regarded as being extraordinarily difficult to answer, with most researchers of the opinion that the ones above are indeed proper!

Each of these complexity classes has *complete languages* with the property that if some complete language for NTIME(*poly*), say, is in DTIME(*poly*) then NTIME(*poly*) = DTIME(*poly*). That is, the difficulty of a complexity-theoretic containment question is entirely encapsulated in the question of whether one particular language is in some complexity class or not. Complexity classes tend to have numerous complete languages and the proof that a language is complete for some complexity class, such as NTIME(*poly*), is usually interpreted as providing strong evidence that the particular complete language is not in (in this case) DTIME(*poly*).

The precise complexity of word problems of groups and classes of groups (in terms of in which complexity classes they lie and whether they are complete for these complexity classes or not) has not been investigated in a systematic fashion. However, some results do exist, amongst them the following: Avenhaus and Madlener [3, 4, 5] showed that a group may have a decidable word problem of varying degrees of space complexity; Lipton and Zalcstein [24] and Simon [34] showed that the word problem for linear groups over fields of zero and prime characteristic, respectively, can be solved in the complexity class DSPACE(*log*); Weispfenning [39] showed that the word problem for *abelian lattice-ordered groups* (an abelian group which is also a lattice such that the group operations are compatible with the lattice operations) is complete for the complexity class co-DTIME(*poly*) (the complement of non-deterministic polynomial-time); Garzon and Zalcstein [15] showed that if certain space complexity classes (containing DSPACE(*log*)) can be separated at all then there are separating word problems of certain Grigorchuk groups witnessing this separation; and Immerman and Landau [21] exhibited a number of groups for which the word problem is complete for complexity classes "contained in" DTIME(*poly*). Surprisingly, there seems to be no group or class of groups for which the word problem is known to be complete for DTIME(*poly*) or NTIME(*poly*) (see [14, 38], for example).

A strong link between automatic groups and computational complexity is that the word problem for automatic groups can be solved in $O(n^2)$ time; moreover, the word problem for word-hyperbolic groups (i.e., groups for which Dehn's algorithm solves the word problem) can be solved in $O(n)$ time (when the Turing machine is allowed to have at least two work tapes: see [11]). This leads to the question (as in [33]) as to whether there is a natural characterization of the class of groups whose word problem can be solved in $O(n)$ time (respectively $O(n^2)$) time. Also, might the word problem for automatic groups be complete for DTIME(*poly*)?

Apart from those hinted at above, other directions for research in this general area present themselves. For example, there has been an explosion of interest in the model theory of finite structures in recent years, especially in relation to complexity theory. In a private communication, Grädel claims to have a purely model-theoretic characterization of automatic groups, and work has already begun (involving the authors and the Helsinki Logic Group) on classifying (word) problems in group theory according to whether they can be defined in certain logics or not.

Acknowledgements. We are grateful to Sarah Rees for several helpful and constructive comments on an early version of this paper. The second author would also like to thank Hilary Craig for all her help and encouragement.

References

[1] A. V. Anisimov, Group languages, *Kibernetika* **4** (1971) 18–24.

[2] A. V. Anisimov, Some algorithmic problems for groups and context-free languages, *Kibernetika* **8** (1972) 4–11.

[3] J. Avenhaus and K. Madlener, Subrekursive Komplexität bei Gruppen I, *Acta Informat.* **9** (1977) 87–104.

[4] J. Avenhaus and K. Madlener, Subrekursive Komplexität bei Gruppen II, *Acta Informat.* **9** (1978) 183–193.

[5] J. Avenhaus and K. Madlener, Algorithmische Probleme bei Einrelatorgruppen und ihre Komplexität, *Arch. Math. Logic* **19** (1978) 3–12.

[6] G. Baumslag, S. M. Gersten, M. Shapiro and H. Short, Automatic groups and amalgams, *J. Pure Appl. Algebra* **76** (1991) 229–316.

[7] J. Berstel, *Transductions and Context-Free Languages*, Teubner (1979).

[8] W. W. Boone, The word problem, *Ann. Math.* **70** (1959) 207–265.

[9] W. W. Boone and G. Higman, An algebraic characterization of groups with a solvable word problem, *J. Australian Math. Soc.* **18** (1974) 41–53.

[10] J. W. Cannon, The combinatorial structure of cocompact discrete hyperbolic groups, *Geom. Dedicata* **16** (1984) 123–148.

[11] B. Domanski and M. Anshel, The complexity of Dehn's algorithm for word problems in groups, *J. Algorithms* **6** (1985) 543–549.

[12] M. J. Dunwoody, The accessibility of finitely presented groups, *Invent. Math.* **81** (1985) 449–457.

[13] D. B. A. Epstein, J. W. Cannon, D. F. Holt, S. Levy, M. S. Patterson and W. Thurston, *Word Processing in Groups*, Jones and Bartlett, London/Boston (1992).

[14] M. Garzon and Y. Zalcstein, On isomorphism testing of a class of 2-nilpotent groups, *J. Comput. System Sci.* **42** (1991) 237–248.

[15] M. Garzon and Y. Zalcstein, The complexity of Grigorchuk groups with application to cryptography, *Theoret. Comput. Sci.* **88** (1991) 83–98.

[16] M. A. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley (1978).

[17] T. Herbst, On a subclass of context-free groups, *Theor. Informatics and Applications* **25** (1991) 255 –272.

[18] T. Herbst, Some remarks on a theorem of Sakarovitch, *J. Comput. System Sci.* **44** (1992) 160–165.

[19] T. Herbst and R. M. Thomas, Group presentations, formal languages and characterizations of one-counter groups, *Theoret. Comput. Sci.* **112** (1993) 187–213.

[20] J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley (1979).

[21] N. Immerman and S. Landau, The complexity of iterated multiplication, *Inform. Computat.* **116** (1995) 103–116.

[22] O. Kharlampovich, A finitely presented soluble group with unsoluble word problem, *Izvestia Akad. Nauk. Ser. Math* **45** (1981) 852–873.

[23] A. A. Letičevskiĭ and L. B. Smikun, On a class of groups with solvable problem of automata equivalence, *Dokl. Acad. Nauk SSSR* **227** (1976); translated in *Soviet Math. Dokl.* **17** (1976) 341–344.

[24] R. J. Lipton and Y. Zalcstein, Word problems solvable in Logspace, *J. Assoc. Comput. Mach.* **24** (1977) 522–526.

[25] R. C. Lyndon and P. E. Schupp, *Combinatorial Group Theory*, Ergebnisse der Mathematik und ihrer Grenzgebiete **89**, Springer-Verlag (1977).

[26] W. Magnus, A. Karrass and D. Solitar, *Combinatorial Group Theory*, Dover Publications (1976).

[27] C. F. Miller, Decision problems for groups - a survey, in: G. Baumslag and C. F. Miller (eds.), *Algorithms and Classification in Combinatorial Group Theory*, MSRI Publications **23**, Springer Verlag (1992) 1–59.

[28] D. E. Muller and P. E. Schupp, Groups, the theory of ends and context-free languages, *J. Comput. System Sci.* **26** (1983) 295–310.

[29] D. E. Muller and P. E. Schupp, The theory of ends, pushdown automata, and second-order logic, *Theor. Comp. Sci.* **37** (1985) 51–75.

[30] P. S. Novikov, On the algorithmic unsolvability of the word problem in group theory, *Trudy. Mat. Inst. Steklov* **44** (1955) 1–143.

[31] C. H. Papadimitriou, *Computational Complexity Theory*, Addison-Wesley (1995).

[32] M. Shapiro, A note on context-sensitive languages and word problems, *Int. J. Alg. Computat.* **4** (1994) 493–497.

[33] H. Short, An introduction to automatic groups, in: J. Fountain (ed.), *Semigroups, Formal Languages and Groups*, NATO ASI Series **C466**, Kluwer (1995) 233–253.

[34] H.-U. Simon, Word problems for groups and context-free recognition, in: L. Budach (ed.), *Fundamentals of Computation Theory*, Math. Research **2**, Akademie-Verlag, Berlin (1979) 417–422.

[35] L. B. Smihun, Relations between context-free groups and groups with a decidable problem of automata equivalence, *Kibernetica* **5** (1976) 33–37.

[36] J. Stallings, *Group Theory and Three-dimensional Manifolds*, Yale University Press (1971).

[37] R. J. Thompson, Embeddings into finitely generated groups which preserve the word problem, in: S. I. Adian, W. W. Boone and G. Higman (eds.), *Word Problems II*, North Holland, Amsterdam (1980).

[38] C. Tretkoff, Complexity, combinatorial group theory and the language of palutators, *Theoret. Comput. Sci.* **56** (1988) 253–275.

[39] V. Weispfenning, The complexity of the word problem for abelian l-groups, *Theoret. Comput. Sci.* **48** (1986) 127–132.