J. London Math. Soc. (2) 67 (2003) 289–301 © 2003 London Mathematical Society
DOI: 10.1112/S0024610702003770

ON REAL-TIME WORD PROBLEMS

DEREK F. HOLT AND CLAAS E. RÖVER

Abstract

It is proved that the word problem of the direct product of two free groups of rank 2 can be recognised by a 2-tape real-time but not by a 1-tape real-time Turing machine. It is also proved that the Baumslag–Solitar groups $B(1,r)$ have the 5-tape real-time word problem for all $r \neq 0$.

1. *Introduction*

For a finite alphabet A, a language $L \subseteq A^*$ is said to be *real-time recognisable* (or just real-time, for short), if it is recognisable by an n-tape deterministic real-time Turing machine for some integer $n \geqslant 0$. Such a machine consists of a tape containing the input word with the tape-head initially at the beginning of the word, together with n work-tapes, which are two-way infinite tapes and initially empty. It has a finite number of states, including a unique initial state and a collection of accepting states. On every transition it must read one input symbol and the input tape-head then moves one place to the right. At the same time, each of the work-tapes may write one symbol or a blank in its current location from a fixed finite set S of work-symbols, and then move at most one square to the left or right, depending on the state, the input symbol, and the symbols currently being scanned by the work-tapes. We shall call this process of writing a symbol and then possibly moving one square left or right a *basic move* on that work-tape. The word is accepted if the machine is in an accepting state on reaching the end of input.

Consider an apparently more general class of languages in which, for some fixed positive integer k, the work-tapes are allowed to perform up to k basic moves for each input symbol read. It is proved in [**6**, Theorem 2] that for any language L in this class, by increasing the number of states and work-symbols, we can construct a standard real-time Turing machine with the same number of work-tapes that recognises L.

An example is given in [**11**] of a language that is acceptable by a 2-tape real-time Turing machine but not by a 1-tape machine. Then, in [**1**], examples are given of languages that are accepted by k-tape but not by $(k-1)$-tape real-time Turing machines for each $k > 1$.

Now let $G = \langle U \rangle$ be a finitely generated group, let $A = U \cup U^{-1}$, and let A^* be the set of all words in A. The *word problem* $W_U(G)$ of G with respect to U is defined to be the set $\{u \in A^* \mid u =_G 1\}$.

The problem of which finitely generated groups G have real-time word problem was investigated in [**7**]. In particular, it was proved that this property is independent of the choice of generators of G, and various closure properties were established,

Received 31 October 2001; revised 9 May 2002.

2000 *Mathematics Subject Classification* 03D40.

The second author is supported by an EPSRC research grant.

such as closure under direct products, passing to finitely generated subgroups, and passing to supergroups in which the original group has finite index. By using the remark above about allowing k work-tape operations for each input symbol read, it is easy to show that, for any positive integer n, the property of $W_U(G)$ being recognisable by a real-time Turing machine with n work-tapes is also independent of the choice of generators of G, and so it is a property of the group G alone. Let us say that groups with this property have n-tape real-time word problem. It can also be easily shown that this class of groups is closed under passing to finitely generated subgroups and passing to supergroups in which the original group has finite index. However, our example in Section 4 will show that it is not closed under direct products, at least in the case $n = 1$.

The main result of [**7**] is that various groups, including finitely generated virtually nilpotent groups, word-hyperbolic groups and geometrically finite hyperbolic groups have 4-tape real-time word problem. (Recall that, for a property $\mathscr{P}$, a group is said to be *virtually* $\mathscr{P}$ if it has a subgroup of finite index with property $\mathscr{P}$.) In fact it was shown that these groups have *tidy* 4-tape real-time word problem, which means that all work-tapes are empty at the end of input of any accepted word. For non-zero integers r and s, the Baumslag–Solitar group $B(s,r)$ is defined by the presentation $B(s,r) = \langle\, x, y \mid y^{-1}x^s y = x^r \,\rangle$. Our first result in this paper is that these groups have tidy 5-tape real-time word problem when $s = 1$. We do not know whether they have real-time word problem when $s \neq 1$.

A group has 0-tape real-time (that is, regular) word problem if and only if it is finite. It is proved in [**9**, **10**] that $W_U(G)$ is (deterministic or non-deterministic) context-free if and only if G is virtually free. (Note that this proof makes use of a deep result of Dunwoody [**3**].) Now a one-stack pushdown automaton is a 1-tape Turing machine which is more restrictive in one sense than a 1-tape real-time machine, because it can only access the top of the stack, but less restrictive in another sense, because it need not read its input word at constant rate. It is easy to show that finitely generated free and hence also virtually free groups have 1-tape real-time word problem. However, it is proved in [**4**] that finitely generated abelian, and hence also virtually abelian, groups have 1-tape real-time word problem, so the class of groups with context-free word problem is strictly contained in that of groups with 1-tape real-time word problem. We shall show in this paper that the wreath product of any non-trivial finite group with an infinite cyclic group has 1-tape real-time word problem.

Finally, we use similar ideas to those in [**11**] to show that the group $F_2 \times F_2$ has 2-tape but not 1-tape real-time word problem, where F_2 is the free group of rank 2. It would be interesting to know whether or not $F_1 \times F_2$ has 1-tape real-time word problem.

2. *Baumslag–Solitar groups*

In this section, we shall prove the following theorem.

THEOREM 2.1. *Let r be a non-zero integer, and let G be the Baumslag–Solitar group $B(1,r) = \langle\, x, y \mid y^{-1}xy = x^r \,\rangle$. Then G has tidy 5-tape real-time word problem.*

Note that $B(1,1)$ is free abelian of rank 2, and $B(1,-1)$ has a free abelian subgroup $\langle x, y^2\rangle$ of index 2. In these cases, G has 1-tape real-time word problem by the result proved in [**4**], and it can easily be seen directly that G has tidy 2-tape real-time word problem. We shall therefore assume from now on that $|r| > 1$.

We denote x^{-1} and y^{-1} by X and Y, respectively, and let $A = \{x, y, X, Y\}$. By [**8**, Theorem 2.1, Chapter IV], any element $g \in G$ can be written in a unique way in the normal form

$$y^l x^{i_1} Y x^{i_2} Y \dots x^{i_m} Y x^n \qquad (1)$$

where l and m are non-negative integers, n is any integer, $0 \leqslant i_j < |r|$ for $1 \leqslant j \leqslant m$, and $i_1 > 0$ if $l, m > 0$.

Let us consider what happens to the normal form when we multiply g on the right by an element of A. When we multiply by x or X, we simply increase or decrease n by 1.

When we multiply by y then, if $m = 0$ we increase l by 1 and replace n by rn, and if $m > 0$ we decrease m by 1 and replace n by $rn + i_m$.

Let $n = rn' + t$ with $0 \leqslant t < |r|$. When we multiply by Y then n is replaced by n'. In addition, if $m = t = 0$ and $l > 0$, then l is decreased by 1, and otherwise m is increased by 1 and i_m is put equal to t.

It is easy to see that, for a given word-length $t + 1$, the words xy^t and Xy^t in A^* (and also $x^2 y^{t-1}$ and $X^2 y^{t-1}$ when $|r| = 2$) define group elements with the largest possible value of $|n|$ in the normal form (1), namely $|n| = |r|^t$. We therefore have the following lemma.

Lemma 2.2. *Suppose $g \in G$ has normal form as in* (1) *where $n \neq 0$. Then any word in A^* representing g has length greater than* $\log_{|r|} |n|$.

We shall now describe a 5-tape real-time Turing machine with language equal to $W_{\{x,y\}}(G)$. The set S of work-symbols will contain an end-of-tape symbol $\mathtt{E}$, the generators y, Y, and symbols x_i and X_i, for $1 \leqslant i < |r|$, that represent x^i and X^i, respectively. We shall use $\mathtt{B}$ to represent a blank cell on a tape.

Two of the five tapes, which we shall call the y-tape and the x-tape, will be used to store the words $y^l x^{i_1} Y x^{i_2} Y \dots x^{i_m} Y$ and x^n, respectively, in the normal form of the group element g defined by the prefix of the input word that has been processed so far. We shall refer to the procedure of adjusting the x- and y-tapes as a result of multiplying g on the right by a generator $z \in A$ as *processing* z.

Two of the other three tapes are called *queueing* tapes, and play the same rôle as in the real-time Turing machine described in [**7**, Section 4]. The fifth tape is called the *reversing* tape. As we shall see shortly, there is no uniform bound on the number of basic moves needed to adjust the normal form stored on the x-tape when processing a single generator. Since we are obliged to read generators from the input tape at a constant rate, we sometimes need to queue the input for later processing. We do this on one of the two queueing tapes. When we have finished the adjustments to the normal form, we first transfer the queued generators onto the reversing tape, thereby restoring them to their original order, and then process them. These generators are deleted from the queueing and reversing tapes as they are transferred or processed. While we are doing all that, we use the other queueing tape to queue generators read from the input tape.

An input word is accepted if all five work-tapes are empty on reaching the end of input.

The word on the y-tape is stored just as it is, except that the symbol x_{i_j} is used in place of x^{i_j} when $i_j > 0$, so this word has length at most $2m + l$. The tape-head is kept pointing at the final symbol at the right-hand end of the word. From the description above of the effect on the normal form when we multiply on the right by an element of A, we see that at most three basic moves are required to adjust the contents of the y-tape when we process a generator from A. (Note that moving the tape-head and then writing amounts to two basic moves, not one!)

Let us assume for the moment that $r > 0$. The changes necessary when $r < 0$ are not great, and will be easy to describe later.

The word x^n on the x-tape is stored in compressed form, similar to that described for nilpotent groups in [**2**] and used in the real-time Turing machine in [**7**]. The word is stored as a sequence of symbols $\sigma_{t-1}\sigma_{t-2}\dots\sigma_0$, where each σ_k is equal to x_{j_k} or X_{j_k} for some j_k, or to $\mathtt{B}$. We define j_k to be 0 when $\sigma_k = \mathtt{B}$. The group element that this represents is defined to be x^n with $n = \sum_{k=0}^{t-1} \epsilon_k j_k r^k$, where $\epsilon_k = 1$ or -1 when $\sigma_k = x_{j_k}$ or X_{j_k}, respectively.

Since powers of both x and X are allowed in the word, the expression for a given element x^n is not unique. For example, if $r = 2$, then $x_1\mathtt{BB}X_1$ and $x_1x_1x_1$ are both legitimate strings representing x^7. However, we do make one extra restriction, and require that no pair of adjacent symbols $\sigma_{k+1}\sigma_k$ should be equal to x_1X_j or to X_1x_j for any $j > 0$. Whenever such a pair does occur, we immediately replace it by the equivalent pair $\mathtt{B}x_{r-j}$ or $\mathtt{B}X_{r-j}$, respectively. Let us call this type of replacement a *simplification.*

Since the word on the x-tape may need to expand or contract at either end, the x-tape is a two-way infinite tape, and we keep end-of-tape symbols $\mathtt{E}$ to the left of σ_{t-1} and to the right of σ_0. After processing each input generator, the tape-head is returned to point at the end-of-tape symbol $\mathtt{E}$ at the right-hand (least significant) end of the word.

We define the length l_x of the word on the x-tape to be equal to the number of non-blank symbols σ_i in the word. Suppose the word stored on the x-tape is x^n or X^n with $n \geqslant 0$. What is the smallest possible value of n for a given value of l_x? For $l_x = 0$ or 1 this is clearly equal to l_x. When $l_x > 1$, bearing in mind that we cannot have x_1X_j or X_1x_j on the tape, we see that the smallest value of n occurs for $r > 2$ when $t = l_x$, $\sigma_{t-1} = x_2$, and $\sigma_k = X_{r-1}$ for $0 \leqslant k < t-1$, and for $r = 2$ when $t = l_x + 1$, $\sigma_{t-1} = x_1$, $\sigma_{t-2} = \mathtt{B}$ and $\sigma_k = X_1$ for $0 \leqslant k < t-2$. In either case, the smallest value of n is equal to $r^{l_x-1} + 1$. Lemma 2.2 now implies the following.

LEMMA 2.3. *Suppose that $g \in G$ has normal form as in* (1) *and is stored on the x- and y-tapes in the manner described above. Then any word in A^* representing g has length at least l_x.*

We now need to consider the adjustment that needs to be made to the x-tape when a generator is processed. If the generator is Y, then we move the tape-head one place to the left, and then replace the current σ_0 by $\mathtt{E}$. If the generator is y, then we write the symbol x_{i_m} (or $\mathtt{B}$ if $m = 0$ or $i_m = 0$) that was deleted from the y-tape in the current position; this becomes the new σ_0. We may now need to make a simplification to the new pair $\sigma_1\sigma_0$. Finally, we move the tape-head one place to the right and write $\mathtt{E}$ there. The number of basic moves on the x-tape required to make these adjustments is at most four if a simplification is needed, and at most two otherwise. When the input generator is x or X, the number of tape-head moves required to make the adjustment is not bounded uniformly. This is why we need the queueing tapes. Suppose that the input symbol is x. (When it is X, simply interchange x and X in the following description.) There is an integer

$j \geqslant 0$ such that $\sigma_i = x_{r-1}$ for $0 \leqslant i < j$ and $\sigma_j \neq x_{r-1}$. Then σ_i must be replaced by B for $0 \leqslant i < j$, and σ_j must be replaced by the symbol that represents $\sigma_j x$. A simplification may now be necessary on $\sigma_{j+1}\sigma_j$. It may also be necessary to write the end-of-tape symbol E to the left of σ_j. To carry out these changes, the tape-head on the x-tape needs to be moved up to $j+3$ places to the left and then back again at the end, making a total of up to $2(j+3)$ basic moves.

Suppose that our Turing machine is a k-real-time machine; that is, up to k basic moves on the work-tapes may be made between reading generators from the input tape. We will specify an appropriate value of k shortly.

The machine starts off in what we shall call *normal* mode, in which generators are read from the input tape and processed immediately. Suppose that the machine is in normal mode, and a generator x or X is read from the input tape when $\sigma_0 = x_{r-1}$ or X_{r-1}, respectively. We shall call this generator x or X *problematic*. Let l_0 be the value of l_x immediately before it is read. If $l_0 = 1$, then the x-tape can be updated with at most six tape-head moves, so provided that we choose $k \geqslant 6$, we can remain in normal mode. If $l_0 > 1$, however, then the machine goes into *queueing* mode. It returns to normal mode when the problematic generator and all generators queued while in queueing mode have been processed.

The only situation in which this machine could return an incorrect answer would be if the input word was equal to the identity, and we reached the end of input while in queueing mode. We need to prove that this cannot happen.

The argument that follows is very similar to the proof of [**7**, Theorem 4.1]. Each queueing mode is divided into phases. During the first phase, the problematic generator is processed, while input generators are queued as necessary on a queueing tape. In each subsequent phase, the generators on the current queueing tape are transferred to the reversing tape, and then they are processed. Queueing mode ends when the current queueing tape is still empty at the end of a phase.

Suppose that, for $i \geqslant 1$, r_i generators are read and queued from the input tape during phase i. These r_i symbols are processed during phase $i+1$. We put $r_0 = 1$, because only the problematic symbol is processed during phase 1. Let $\rho_i = r_0 + r_1 + \ldots + r_i$ for $i \geqslant 0$. If we can prove that $\rho_j < l_0$ for some $j \geqslant 0$, then it will follow from Lemma 2.3 that, during the first j phases of the queueing mode, the elements of G represented by the prefixes of the input word that have been read so far from the input tape cannot be the identity. Hence, if we can prove it for all j, then it will show that if we run out of input during queueing mode, then the input word cannot equal the identity in G.

We prove that $\rho_j < l_0$ by induction on j. It is true for $j = 0$, because we are assuming that $l_0 > 1$. Suppose that, for some j, we know that $\rho_{j-1} < l_0$, and hence that l_x remains greater than 0 throughout the first j phases.

As we saw above, when a y or Y is processed, either at most two basic moves are made on the x-tape and l_x is increased by at most 1, or four basic moves are made, including a simplification, and l_x remains constant.

When an x or X is processed then, if a sequence of substitutions is provoked, almost all of the ensuing moves of the x-tape-head to the left result in l_x being decreased by 1. Since the total number of basic moves is equal to twice the number of tape-head moves to the left, we find that in all cases, if a total of t basic moves are made on the x-tape, then l_x is decreased by at least $t/2 - 3$.

Hence, whatever generator is processed, it is true that if t basic moves are made on the x-tape, then l_x is decreased by at least $t/2 - 3$. Suppose that t_i basic moves are made on the x-tape during the ith phase, and let $\tau_i = t_1 + \ldots + t_i$. Then, during the first i phases, l_x is decreased by at least $\tau_i/2 - 3\rho_{i-1}$ and since, by inductive hypothesis, $l_x > 0$ throughout the first j phases, we must have

$$\tau_j/2 - 3\rho_{j-1} < l_0. \tag{2}$$

Since each phase except the first begins with the r_{i-1} basic moves needed to transfer symbols from the queueing tape to the reversing tape, we have $kr_i \leqslant t_i + r_{i-1}$ for $i > 0$, and summing from $i = 1$ to j gives $k(\rho_j - 1) \leqslant \tau_j + \rho_{j-1}$. Combining this equation with (2) yields $k(\rho_j - 1) \leqslant 2l_0 + 7\rho_{j-1}$ and so using our inductive hypothesis that $\rho_{j-1} < l_0$, we get $k(\rho_j - 1) < 9l_0$. Thus, since we are assuming that $l_0 > 1$, we get $\rho_j < l_0$ provided that $k \geqslant 18$. This completes the induction and the proof of Theorem 2.1 in the case $r > 0$.

When $r < 0$, the only difference is that when a y or Y is processed, the element of $\langle x \rangle$ represented by the word stored on the x-tape needs to be inverted. Rather than change every x_i to an X_i and X_i to an x_i, which would take too much time, we simply double the number of states of the machine, and introduce 1 or -1 as an extra component of each state. When this component is -1 it means that each x_i on the x-tape now represents X^i, and each X_i represents x^i.

3. *Wreath products*

Let A and C be groups. The *restricted wreath product* $G := A \wr C$ of A with C is defined as follows. Let B be the set of all functions $b : C \longrightarrow A$ with $b(c) = 1$ for all but finitely many $c \in C$. Then, under coordinate-wise multiplication, B is isomorphic to the direct sum of $|C|$ copies of A, and G is the semi-direct product of B with C where the action of C is given by

$$b^c(c') = b(c'c^{-1}). \tag{3}$$

Theorem 3.1. *The restricted wreath product of an arbitrary finite group with an infinite cyclic group has tidy* 1*-tape real-time word problem.*

Proof. Let A be a finite group, and $G = A \wr \langle t \rangle = B \rtimes \langle t \rangle$. For $a \in A$ define $b_a \in B$ by

$$b_a(t^i) = \begin{cases} a & \text{if } i = 0 \\ 1 & \text{otherwise.} \end{cases} \tag{4}$$

Then G is generated by $A_0 = \{b_a \mid a \in A\}$ together with t, and every element g of G is of the form $g = bt^i$ with $b \in B$ and $i \in \mathbb{Z}$. Hence $gt^{\pm 1} = bt^{i \pm 1}$ and $gb_a = bb_a^{t^{-i}}t^i$, and, by (4) and (3),

$$(bb_a^{t^{-i}})(t^j) = \begin{cases} b(t^j) & \text{if } j \neq -i \\ b(t^j)a & \text{if } j = -i. \end{cases}$$

Now g is the identity in G if and only if $i = 0$ and $b(t^j) = 1$ for all $j \in \mathbb{Z}$. Thus, if we interpret $b(t^j)$ as the content of the jth square on a two-way infinite tape and t^i as 'the tape-head scans the $-i$th square of the tape', then multiplication by a generator can be carried out by a single basic move of a 1-tape Turing machine M with work-symbols A. If we let the identity of A be represented by the blank symbol, then, in order to accept the word problem of G, M has to accept an input word if and only if its work-tape is empty and the tape-head is scanning the 0th square after the input has been read. Using an end-of-tape symbol E and another disjoint copy of A, say $\bar{A}$, whose symbols appear only in the 0th square which is now banned for A-letters, it is a routine matter to turn M into a proper tidy 1-tape real-time Turing machine accepting the word problem of G. □

It is well known that, unless A is trivial, G is not finitely presentable. Hence we have the following.

Corollary 3.2. *The class of groups with tidy* 1*-tape real-time word problem contains groups that are not finitely presentable.*

As far as we are aware this is the first example of a group which is not finitely presentable with 1-tape real-time word problem. It is well known that the direct product of two free groups of rank 2 contains finitely generated subgroups that are not finitely presentable, which provides a source of such examples with 2-tape real-time word problem.

4. *Direct products of free groups*

This section is devoted to the proof of the following result.

Theorem 4.1. *The word problem of a direct product of two free groups of rank* 2 *is not a* 1*-tape real-time language.*

In fact we shall prove a stronger result. Let G be a finitely generated group with finite generating set $A = A^{-1}$. For each positive integer n, let $s_{G,A}(n)$ be the number of elements of G that can be represented by a word over A of length n but not by a word of length less than n. Alternatively, $s_{G,A}(n)$ can be viewed as the size of the sphere of radius n in the Cayley graph of G with respect to A. Then G is said to have *exponential growth* if there is a constant K such that, for all $n \geqslant 1$, $s_{G,A}(Kn) \geqslant 2^n$. It is easy to see that this definition is independent of the choice of A and that non-abelian free groups have exponential growth. Hence, Theorem 4.1 follows immediately from the following result.

Theorem 4.2. *The word problem of a direct product of two groups of exponential growth is not a* 1*-tape real-time language.*

Theorem 4.2 has the following corollary, by the remarks in the introduction about the closure properties of the class of groups with n-tape real-time word problem.

Corollary 4.3. *The word problem of a group which has a subgroup isomorphic to the direct product of two groups of exponential growth is not a* 1*-tape real-time language.*

Since non-elementary hyperbolic groups have exponential growth (see for example **[5]**) we have the following result.

COROLLARY 4.4. *The word problem of the direct product of two non-elementary hyperbolic groups is not a* 1*-tape real-time language.*

We also get the following result about complexity classes of word problems.

COROLLARY 4.5. *The class of groups with* 1*-tape real-time word problem is a proper subclass of the class of groups with* 2*-tape real-time word problem.*

4.1. *Proof of Theorem* 4.2

From now on let G_i, $i = 1, 2$, be finitely generated groups of exponential growth, and put $G = G_1 \times G_2$. By the remarks in the introduction we may choose any finite generating set for G, so we fix finite generating sets A_i for G_i, $i = 1, 2$, and take $A = A_1 \cup A_2$ as generating set for G. We are interested in the word problem W of G with respect to A.

The proof is by contradiction, so we assume that we have a 1-tape real-time machine M which accepts W. In the sequel, 'tape' always means the work-tape of M. When w is a word over A, then $t(w)$ denotes the tape of M after processing w, by which we mean all tape squares that were visited during that computation together with their contents. The tape $t(w)$ together with the internal state and the current position of the tape-head is called the *configuration* of M after w has been processed, and is denoted by $c(w)$. The *length* of the tape $t(w)$, written $|t(w)|$, is the number of tape squares that were visited while processing w. By definition, this is also the length of the configuration $c(w)$. It is clear that $|t(w)|$ never exceeds the length of w plus one.

The idea behind the proof is to apply the bottleneck argument of Rabin [**11**] twice to find words $u, v \in A_1^*$ and $z \in A_2^*$ such that $uzv^{-1}z^{-1}$ is accepted by M but u and v represent distinct elements of G_1. Here and in the sequel, u^{-1} denotes the inverse of the word u as in free groups; that is, reverse the order of the letters and invert them, for example $(abc^{-1}a)^{-1} = a^{-1}cb^{-1}a^{-1}$.

Let K be a constant satisfying

$$s_{G_i,A_i}(Kn) \geqslant 2^n \qquad \text{for } i = 1, 2. \tag{5}$$

Such a K exists by hypothesis and we lose nothing by assuming that K is a non-negative integer. Let m be a power of 2 greater than the number of internal states of M and also greater than the number of work-symbols of M, where the blank is considered to be a work-symbol. Then $m \geqslant 2$. Clearly there are at most $m^k \cdot k \cdot m$ different configurations of length at most k, and for sufficiently large k, $m^k \cdot k \cdot m \leqslant m^{2k}$ holds. In the sequel logarithms are to the base 2. Define

$$c = \frac{1}{2 \log m}, \quad d = \frac{20K}{c}, \quad \text{and} \quad N = m^2 + m^4 + \ldots + m^{2d}.$$

Notice that $1/c$ and d are integers, by the choice of m and K. It is clearly possible to find an integer $l_0 > 1$ such that for all integers $l \geqslant l_0$

(i) $m^{9(l+1)}9(l+1)m \leqslant m^{2 \cdot 9(l+1)}$;

(ii) $2^l > 2(Kl + 1)N^2(10K(Kl + 1)/c)^{2d}$.

Now fix an integer $l \geqslant l_0$ and put

$$L = Kl, \quad i = \frac{10(L+1)}{c}, \quad I = Ki, \quad \text{and} \quad f = \frac{2^l}{2^l + 1}.$$

Notice that i, I and L are integers, for $1/c$ and K are. Because

$$ci = 9(L+1) + \log 2^{L+1} > 9(L+1) + \frac{\log(2^l + 1)}{2\log m}$$

with these choices we have

$$ci > 9(L+1) - \frac{\log(1-f)}{2\log m} \tag{6}$$

and

$$2^l > 2(L+1)(NI^d)^2. \tag{7}$$

Let U be a set of 2^l words in A_1^* of length L such that distinct elements of U represent distinct elements of G_1, and let V be a set of 2^i words in A_2^* of length I such that distinct elements of V represent distinct elements of G_2. The existence of U and V are guaranteed by (5).

LEMMA 4.6. *For every $u \in U$ there are at least $f2^i$ different $v \in V$ with $|t(uv)| \geqslant 9(L+1)$.*

Proof. By the definition of c and (6), we get

$$(1-f)2^i > m^{2\cdot 9(L+1)}. \tag{8}$$

For a given u and different v_1 and v_2 in V the configurations $c(uv_1)$ and $c(uv_2)$ must also be different because uv_1 and uv_2 represent different elements of the group G. Since $L \geqslant l \geqslant l_0$, by (i) above, the right-hand side of (8) exceeds the number of different configurations of length at most $9(L+1)$, and the lemma follows. □

Together with the fact that the intersection of 2^l subsets of V each of size at least $f2^i$ is of size at least $(1-f)2^i > 1$, and hence not empty, Lemma 4.6 implies the following result.

COROLLARY 4.7. *There exists $w \in V$ such that for all $u \in U$, $|t(uw)| \geqslant 9(L+1)$.*

LEMMA 4.8. *Let $u \in U$ and $v \in V$. Then fewer than $L+1$ of the squares of $t(uv)$ are visited more than d times while processing uv.*

Proof. Since $i > l$, the length of uv is less than $2I$, and so the total number of moves done by M while processing uv is less than $2I$, but if there were at least $L+1$ of the squares of $t(uv)$ visited more than d times, then M needed to do at least $(d+1)(L+1)-1$ moves, that is $\frac{1}{10}(d+1)ci \leqslant 2I$, or equivalently $d+1 \leqslant 20K/c$. This contradicts the choice of d, and the lemma is proved. □

From now on u will always denote an element of U and w will be an element whose existence is established in Corollary 4.7; in particular $|t(uw)| \geqslant 9(L+1)$. Divide the tape $t(uw)$ into three parts X, $t(u)$ and Y, as in Figure 1. In the figures labels like $t(u)$ indicate those tape squares visited while processing u no matter whether their contents are changed later in the computation. Since $|t(u)| \leqslant L+1 \leqslant \frac{1}{9}|t(uw)|$, for at least half of the $u \in U$ one of the parts X or Y of $t(uw)$ contains at least $\frac{4}{9}$ of the tape squares of $t(uw)$. For definiteness assume it is Y, and let U_0 be the set of such u's, so $|U_0| \geqslant |U|/2$.

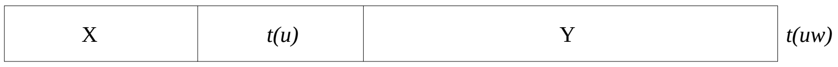

FIGURE 1.

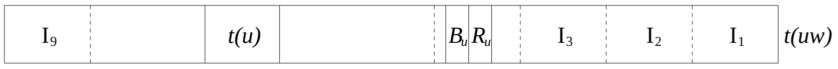

FIGURE 2.

Let $u \in U_0$ and divide the tape $t(uw)$ into nine equally long parts (as accurately as possible), labelled $\mathrm{I}_1, \ldots, \mathrm{I}_9$ from right to left. Since each I_j comprises at least $L+1$ tape squares, by Lemma 4.8, there is a square B_u in part I_4 which is visited at most d times during the processing of uw. Fix such a B_u and denote by R_u the square to the right of B_u (see Figure 2).

Now the pair (B_u, R_u) defines a factorisation $w_0 w_1 \ldots w_r$ of w by the following conditions. The tape-head of M moves from B_u to R_u if and only if the last letter of a w_j with even $j < r$ is being processed, and it moves from R_u to B_u if and only if the last letter of a w_j with odd $j < r$ is being processed.

The *crossing sequence* of w with respect to (B_u, R_u) is the r-tuple

$$((q_0, p_0), \ldots, (q_{r-1}, p_{r-1}))$$

where q_j is the internal state and p_j is the scanned symbol on the tape in the configuration $c(uw_0 \ldots w_j)$, $0 \leqslant j \leqslant r-1$. In particular, the tape-head in the configuration $c(uw_0 \ldots w_j)$, $0 \leqslant j \leqslant r-1$, is scanning B_u or R_u according to whether j is odd or even, respectively. In the sequel a phrase like 'the tape to the right of the square S' means all tape squares to the right of but not including S.

Notice that the number r does not exceed d and hence there are at most N possible crossing sequences and not more than $\binom{I}{d} \leqslant I^d$ different factorisations of w. Thus there is a subset U_1 of U_0 of size at least $|U_0|/(NI^d)$ so that, for all $u \in U_1$, the computations of uw give exactly the same crossing sequences and factorisations of w with respect to (B_u, R_u). Moreover, the tapes $t(uw)$ are all identical to the right of B_u and, for all prefixes z of w for which the tape-head is to the right of B_u in the configuration $c(uz)$, the internal states of the configurations $c(uz)$, $u \in U_1$, are all equal. This is because the part of the tape to the right of B_u, Y say, is changed only while w_j with odd j are being processed and for each such w_j the change depends solely on the pair (q_{j-1}, p_{j-1}) and the previous contents of Y which are the same for all $u \in U_1$ because all factorisations of w are the same and $t(u)$ is to the left of B_u.

Let z be the prefix of w such that the right-most square of $t(uw)$, say E, is occupied for the first time precisely after the processing of uz for some, and hence for all $u \in U_1$. The following follows from this discussion.

LEMMA 4.9. *For all $u \in U_1$, the computations of uz give exactly the same crossing sequences and factorisations of z with respect to (B_u, R_u). Moreover, all of the tapes $t(uz)$ are identical to the right of B_u and the internal states of the configurations $c(uz)$ are also all equal.*

Align all of the tapes $t(uz)$ for $u \in U_1$ at their (right-most) squares E, and let J denote the interval of those $L+1$ tape squares such that there are precisely $L+2$

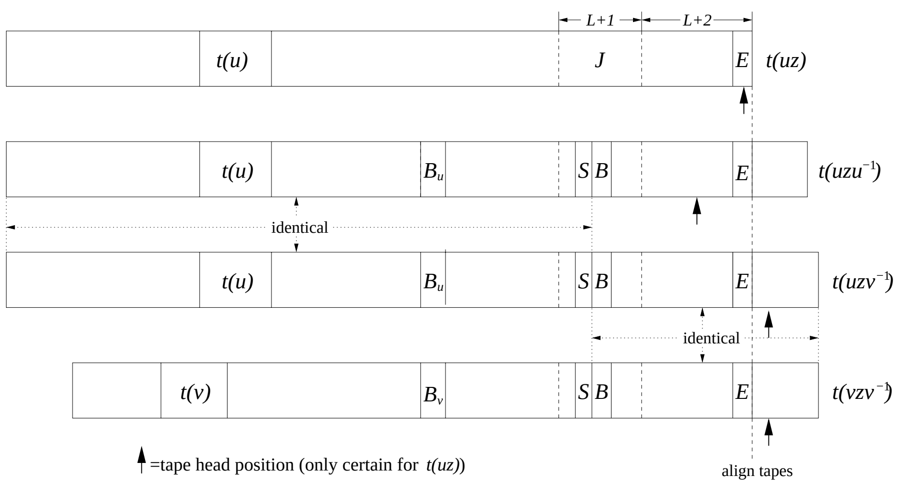

FIGURE 3.

tape squares of $t(uz)$ to the right of J (see Figure 3). Observe that B_u is always to the left of J.

Next consider the tapes $t(uzu^{-1})$, $u \in U_1$. Since u is of length L, the tape-head of $c(uzu^{-1})$ must be to the right of J. Similarly to the proof of Lemma 4.8 one sees that there is a square, C_u say, inside J which is visited at most d times during the computation of z^{-1} that starts in the configuration $c(uzu^{-1})$.

Since there are at most $L+1$ possibilities for C_u, there is a subset U_2 of U_1 satisfying $|U_2| \geqslant |U_1|/(L+1)$ and a fixed square B in J such that for all $u \in U_2$, B is visited at most d times during the computation of z^{-1} which starts in the configuration $c(uzu^{-1})$. Let S denote the square to the left of B. Just as before this gives rise to a factorisation of z^{-1} and an associated crossing sequence from the computation of z^{-1} starting in the configuration $c(uzu^{-1})$, only this time by considering moves from B to S and moves from S to B.

Again there are at most NI^d possible combinations of such factorisations and crossing sequences. Since by (7),

$$|U_2| \geqslant \frac{|U_1|}{L+1} \geqslant \frac{|U_0|}{(L+1)NI^d} \geqslant \frac{2^l}{2(L+1)NI^d} > NI^d,$$

we get the following.

LEMMA 4.10. *There exist $u, v \in U_2$, $u \neq v$, such that the computations of z^{-1} starting in $c(uzu^{-1})$ and $c(vzv^{-1})$ both give the same factorisation of z^{-1}, $z^{-1} = z_0z_1 \dots z_s$ say, as well as the same crossing sequence, say $((q_0, p_0), \dots, (q_{s-1}, p_{s-1}))$, with respect to (S, B).*

The final contradiction in the proof of Theorem 4.2 is now obtained by the following result.

LEMMA 4.11. *Let u and v be distinct elements of U_2 satisfying the conclusion of the previous lemma. Then the configuration $c(uzv^{-1}z^{-1})$ is an accept configuration.*

Proof. The point is that the part of $t(uzv^{-1})$ to the left of B is identical to the part of $t(uzu^{-1})$ to the left of B and that the part of $t(uzv^{-1})$ to the right of S is identical to the part of $t(vzv^{-1})$ to the right of S (see Figure 3).

Let us be a little more precise. When w is a word over A, then $q(w)$ denotes the internal state and $h(w)$ the position of the tape-head in the configuration $c(w)$. For $w \in \{uzu^{-1}, vzv^{-1}, uzv^{-1}\}A_2^*$ let $t_l(w)$ and $t_r(w)$ denote the parts of $t(w)$ to the left of B and to the right of S, respectively. By the choice of u and v, Lemma 4.9, and the discussion following it, the following hold:

$$\begin{aligned} t_l(uzv^{-1}) &= t_l(uzu^{-1}),\\ t_r(uzv^{-1}) &= t_r(vzv^{-1}),\\ q(uzv^{-1}) &= q(vzv^{-1}),\\ h(uzv^{-1}) &= h(vzv^{-1}). \end{aligned} \tag{9}$$

We call a prefix x of z^{-1} even (odd) if the minimal j for which x is a prefix of $z_0 \dots z_j$ is even (odd). We will prove the following.

CLAIM 4.12. *For every prefix x of z^{-1},*

(a) $t_l(uzv^{-1}x) = t_l(uzu^{-1}x)$;

(b) $t_r(uzv^{-1}x) = t_r(vzv^{-1}x)$;

(c) $$q(uzv^{-1}x) = \begin{cases} q(vzv^{-1}x) & \textit{if } x \textit{ is even} \\ q(uzu^{-1}x) & \textit{if } x \textit{ is odd}; \end{cases}$$

(d) $$h(uzv^{-1}x) = \begin{cases} h(vzv^{-1}x) & \textit{if } x \textit{ is even} \\ h(uzu^{-1}x) & \textit{if } x \textit{ is odd}. \end{cases}$$

This completes the proof because M accepts $uzu^{-1}z^{-1}$ and $vzv^{-1}z^{-1}$ solely on the basis of its internal state and the scanned tape symbol at the end of the computation. Also one of these equations agrees, by the claim, with that of $c(uzv^{-1}z^{-1})$ depending on whether z^{-1} is odd or even.

Proof of Claim 4.12. The claim is proved by induction on the length of x, the case when x is the empty word being (9). Let x' and $x = x'a$ be prefixes of z^{-1}, where $a \in A_2$. Suppose that x' satisfies (a)–(d). There are two cases to consider.

Case 1: Both x and x' are even; the case when both are odd is similar.

By the definition of the factorisation, $h(uzv^{-1}x')$ scans a square in $t_r(uzv^{-1}x')$, so (a) holds by induction. Since the action of M is determined by a, $q(uzv^{-1}x')$, and the contents of the square scanned by $h(uzv^{-1}x')$, the induction hypothesis implies that also (b)–(d) hold for x.

Case 2: Suppose that x' is even and x is odd; the opposite case is similar again.

Then $x' = z_0 \dots z_j$ with j even. By the properties of the crossing sequence, the factorisation of z^{-1}, and the induction hypothesis we have

$$q(uzv^{-1}x') = q(vzv^{-1}x') = q(uzu^{-1}x') = q_j,$$

$$h(uzv^{-1}x') = h(vzv^{-1}x') = h(uzu^{-1}x') = S,$$

and S contains p_j. Hence (b) holds by induction, because the part of the tape to the right of S is not changed by the following move of M. (Recall from the introduction that M may only print a symbol on the tape square it is currently scanning.) Statements (a), (c) and (d) follow now from the induction hypothesis, again because the action of M is determined by a, q_j and p_j. $\square$

This now establishes Lemma 4.11 and Theorem 4.2. $\square$

References

1. S. O. Aanderaa, 'On k-tape versus $(k-1)$-tape real time computation', *Complexity of computation*, SIAM-AMS Proceedings VII (American Mathematical Society, Providence, RI, 1974).
2. J. Cannon, O. Goodman and M. Shapiro, 'Dehn's algorithm for non-hyperbolic groups', preprint, University of Melbourne.
3. M. J. Dunwoody, 'The accessibility of finitely presented groups', *Invent. Math.* 81 (1985) 449–457.
4. M. J. Fischer and A. L. Rosenberg, 'Real-time solutions of the origin-crossing problem', *Math. Systems Theory* 2 (1968) 257–263.
5. E. Ghys and P. de la Harpe, *Sur les groupes hyperboliques d'après Mikhael Gromov* (Birkhäuser, Boston, 1990).
6. J. Hartmanis and R. E. Stearns, 'On the computational complexity of algorithms', *Trans. Amer. Math. Soc.* 117 (1965) 285–306.
7. D. F. Holt and S. Rees, 'Solving the word problem in real time', *J. London Math. Soc.* 63 (2001) 623–639.
8. R. C. Lyndon and P. E. Shupp, *Combinatorial group theory* (Springer, Berlin, 1977).
9. D. E. Muller and P. E. Schupp, 'Groups, the theory of ends, and context-free languages', *J. Comput. System Sci.* 26 (1983) 295–310.
10. D. E. Muller and P. E. Schupp, 'The theory of ends, pushdown automata, and second-order logic', *Theoret. Comput. Sci.* 37 (1985) 51–75.
11. M. O. Rabin, 'Real time computation', *Israel J. Math* 1 (1963) 203–211.
12. A. L. Rosenberg, 'Real time definable languages', *J. Assoc. Comput. Mach.* 14 (1967) 645–662.

Derek F. Holt
Mathematics Institute
University of Warwick
Coventry CV4 7AL

dfh@maths.warwick.ac.uk

Claas E. Röver
Department of Mathematics
University of Newcastle-upon-Tyne
Merz Court
Newcastle-upon-Tyne NE2 7RU

c.h.e.w.roever@newcastle.ac.uk