Published in final edited form as:

Int J Algebra Comput. 2008 November 1; 18(7): 1137–1177. doi:10.1142/S0218196708004822.

ON A GENERALIZATION OF DEHN'S ALGORITHM

OLIVER GOODMAN and

Department of Mathematics and Statistics, University of Melbourne, Parkville, Victoria 3052, Australia

MICHAEL SHAPIRO*

Department of Pathology, Tufts University, Boston, MA 02111, USA

OLIVER GOODMAN: ong@optusnet.com.au; MICHAEL SHAPIRO: michael.shapiro@tufts.edu

Abstract

Viewing Dehn's algorithm as a rewriting system, we generalize to allow an alphabet containing letters which do not necessarily represent group elements. This extends the class of groups for which the algorithm solves the word problem to include finitely generated nilpotent groups, many relatively hyperbolic groups including geometrically finite groups and fundamental groups of certain geometrically decomposable 3-manifolds. The class has several nice closure properties. We also show that if a group has an infinite subgroup and one of exponential growth, and they commute, then it does not admit such an algorithm. We dub these Cannon's algorithms.

Keywords

Word problem; rewriting systems; Dehn's algorithm

1. Introduction

1.1. Dehn's algorithm

Early last century Dehn [10] introduced three problems. We know them now as the word problem, the conjugacy problem and the isomorphism problem. Given a finitely generated group G and generating set $\mathcal{g}$, we have solved the word problem if we can give a procedure which determines, for each word $w \in \mathcal{g}^*$ whether or not w represents the identity. We have solved the conjugacy problem if we can give a procedure which determines, for each pair of words $u, v \in \mathcal{g}^*$, whether they represent elements which are conjugate in G . For the isomorphism problem, Dehn invites us to develop procedures for determining if two given groups are isomorphic.

Using hyperbolic geometry Dehn proceeded to solve the word and conjugacy problems for the fundamental groups of closed hyperbolic surfaces. Let us take a moment to describe his solution of the word problem. For specificity, let us take the two-holed surface group

$$\langle x_1, y_1, x_2, y_2 \mid [x_1, y_1][x_2, y_2] \rangle.$$

Communicated by R. J. Grigorchuk

*Michael Shapiro wishes to thank David Thorley-Lawson for support from Public Health Service grant (RO1 AI062989). Additional support for this work came from NIH grant 1K25AI079404-01.

The Cayley graph of this group sits in $\mathbb{H}^2$ as the 1-skeleton of the tessellation of $\mathbb{H}^2$ by regular hyperbolic octagons, and the relator $R = [x_1, y_1][x_2, y_2]$ labels the boundary of each octagon. A word w now lies along the boundaries of these octagons and is a closed curve if and only if it represents the identity. Dehn then shows that any reduced closed curve travels around the far side of some “outermost” octagon and in doing so contains at least 5 of its 8 edges. That is, each reduced word representing the identity contains more than half of a relator. (Here we are allowing cyclic permutations of R and R^{-1} .)

This solves the word problem, for we can decompose the relator as uv^{-1} where u appears in $w = xuy$ and u is longer than v . This allows us to replace w with the shorter word $w' = xvy$. If the word w represents the identity and w' is not empty, we can again shorten w' in similar manner. This process either ends with a non-empty word which we cannot shorten, in which case w did not represent the identity, or with the empty word in which case w did represent the identity.

Accordingly, we say the group G has a *Dehn’s algorithm* if it has a finite presentation

$$\langle G | \mathcal{D} \rangle$$

such that every non-empty word $w \in g^*$ representing the identity contains more than half of some relator in $\mathcal{D}$. Equivalently, we could write $\mathcal{D}$ as a finite set of relations $u_i = v_i$ so that for each i , $\ell(u_i) > \ell(v_i)$ and every word $w \in g^*$ representing the identity contains some u_i .

In [5], Cannon generalized Dehn’s results to groups acting co-compactly, discretely on hyperbolic space. It is a theorem [21,⁶,¹] that a group has a Dehn’s algorithm if and only if it is one of those groups which are variously called Gromov hyperbolic, hyperbolic, negatively curved or word hyperbolic.

1.2. A new definition

Cannon [7] suggested we take the following viewpoint. We have a class of machines designed to carry out^a Dehn’s algorithm. Such a machine would be equipped with a finite set of length reducing replacement rules $u_i \rightarrow v_i$. It would have a window of finite width through which it would examine a given word. This window would start at the beginning of the word. As the window moved along, the machine would scan the word looking for occurrences of u_i ’s. If it fails to find any u_i and is not already at the end of the word, it moves forward. If it finds a u_i it replaces it with the corresponding v_i . (The blank spaces magically evaporate.) The window then moves backwards one letter less than the length of the longest u_i or to the beginning of the word if that is closer. It accepts a word if and only if it succeeds in reducing that word to the empty word.

The key difference here is that our working alphabet is no longer restricted to the group generators. We shall see that there are several different classes of machines here with some rather divergent properties. We do not know if these competing definitions for the title of “Dehn machine” yield different classes of groups. Our most restrictive version solves the word problem in a much larger class of groups than the word hyperbolic groups.

^aMorally, Dehn’s algorithm represents a linear time solution to the word problem, but this actually depends on the machine implementation. If it is implemented on a classical one-tape Turing machine, the running time is $\mathcal{O}(n^2)$ due to the need to exorcize (or traverse) the blanks left by each replacement. If it is implemented on a random access machine, it is $\mathcal{O}(n \log n)$ due to the size of the words needed to indicate addresses. If it is implemented out on a multi-tape machine it is $\mathcal{O}(n)$ since here blanks “evaporate” between the tapes [11]. In [15,¹⁶] it is shown that there is a real-time multi-tape implementation.

We describe these classes of machines in terms of rewritings that they carry out. In each of these, we are supplied with an alphabet $\mathbb{A}$ and a finite set of pairs $(u_i, v_i) \in \mathbb{A}^* \times \mathbb{A}^*$ where for each i , $\ell(u_i) > \ell(v_i)$. We call these *rewriting rules* and write $u_i \rightarrow v_i$. We call u_i and v_i the *left-hand side* and the *right-hand side* respectively. For technical reasons we also have to allow the machines to have *anchored* rules: these are rules which only apply when the left-hand side is an initial segment of the current word. We write $\hat{u}$ for the left-hand side of an anchored rule and consider u and $\hat{u}$ to be distinct.

Let S be a finite set of rewriting rules such that each left-hand side appears at most once. We say that $w \in \mathbb{A}^*$ is *reduced* with respect to S if it contains none of the left-hand sides in S . The following algorithm, which we call the *incremental rewriting algorithm* given by $(\mathbb{A}, S)$, replaces any $w \in \mathbb{A}^*$ by a reduced word in finitely many steps. If w contains a left-hand side, find one which ends closest to the start of w ; if several end at the same letter, choose the longest; if possible, choose an anchored one in preference to a non-anchored one of the same length. Replace it by the corresponding right-hand side. Repeat until w is reduced.

Here is a slightly different definition: the *non-incremental rewriting algorithm* given by $(\mathbb{A}, S)$, replaces any $w \in \mathbb{A}^*$ by a reduced word in finitely many steps. Here S may also include end-anchored rules, with left-hand side $\hat{u}$, and rules anchored at both ends. If w contains a left-hand side, find one which starts closest to the start of w ; if several start at the same letter, choose the longest; prefer anchored rules when there is a choice. Replace it by the corresponding right-hand side. Repeat until w is reduced.

Each of these algorithms gives a reduction map $R = R_S: \mathbb{A}^* \rightarrow \mathbb{A}^*$ where $R(w)$ is the reduced word which the algorithm produces starting with w . The incremental rewriting algorithm gets its name from the following property: if R is the reduction map of an incremental rewriting algorithm, then $R(uv) = R(R(u)v)$. (See Lemma 2.)

We may wish to apply an incremental rewriting algorithm only to words in $\mathbb{A}_0^*$ where $\mathbb{A}_0 \subseteq \mathbb{A}$. We then refer to $\mathbb{A}_0$ as the *input* alphabet and $\mathbb{A}$ as the *working* alphabet. The algorithm can then be given as a triple $(\mathbb{A}_0, \mathbb{A}, S)$. We say that $\{w \in \mathbb{A}_0^* | R(w) \text{ is empty}\}$ is the *language* of this triple. The same can be done for non-incremental rewriting algorithms.^b

Clearly Dehn's algorithm can be implemented as an incremental rewriting algorithm, with $\mathbb{A}_0 = \mathbb{A} = \mathcal{g}$ and S obtained from the u_i . We generalize this as follows. (See Sec. 3 for the example which originally motivated this definition.)

Definition 1—A group G , with semi-group generators $\mathcal{g}$, has a Cannon's algorithm if there exists an alphabet $\mathbb{A} \supseteq \mathcal{g}$, and set of rewriting rules S over $\mathbb{A}$, such that the incremental rewriting algorithm reduces $g \in \mathcal{g}^*$ to the empty word, if and only if g represents the identity in G .

We have chosen incremental rewriting algorithms because of their nice group theoretic properties. Using incremental rewriting algorithms in the previous definition ensures that the Cannon's algorithm remembers group elements. (See Proposition 3.) That is, if G has a Cannon's algorithm with input alphabet $\mathcal{g}$ and reduction map R , and there are x and y in $\mathcal{g}^*$ so that $R(x) = R(y)$, then x and y denote the same element of G . This property does not hold in general if one uses non-incremental rewriting algorithms.

^bSince this work first appeared in preprint form, Kambites and Otto [20] have shown that the incremental rewriting algorithm languages are contained in the set of Church–Rosser languages and that a language is a non-incremental rewriting algorithm language if and only if it is a Church–Rosser language.

On the other hand, non-incremental rewriting algorithms have nice language theoretic properties in that they support composition. In the following, we will conceal some technical details in the word “mimics”. One can imagine the non-incremental rewriting algorithm as being carried out by a machine with a finite number of internal states s_i and a list of rewriting rules S_i for each state s_i . There is a non-incremental rewriting algorithm which mimics the action of this multi-state machine. Consequently, given two non-incremental rewriting algorithms over the same alphabet $\mathbb{A}$ with reduction maps Q and R , there is a non-incremental rewriting algorithm which mimics a non-incremental rewriting algorithm whose reduction map is $R \circ Q$. We will refer to a Cannon’s algorithm carried out using a non-incremental rewriting algorithm as a non-incremental Cannon’s algorithm.

1.3. Results

Before describing our results, we note that many of these were independently rediscovered by Kambites and Otto [20]. We show here that groups with Cannon’s algorithms have the following closure properties:

1. If G has a Cannon’s algorithm over one finite generating set then it has a Cannon’s algorithm over any finite generating set.
2. If G has a Cannon’s algorithm and G is a finite index subgroup of H then H has a Cannon’s algorithm.
3. If G and H have Cannon’s algorithms, then $G * H$ has a Cannon’s algorithm.
4. If G has a Cannon’s algorithm and H is a finitely generated subgroup of G then H has a Cannon’s algorithm.

This last closure property significantly increases the class of groups with Cannon’s algorithms. Every word hyperbolic group has a Cannon’s algorithm, and as Bridson and Miller have pointed out to us, the finitely generated subgroups of word hyperbolic groups include groups which are not finitely presented and groups with unsolvable conjugacy problem [2].

We also show that groups with Cannon’s algorithms include

1. finitely generated nilpotent groups,
2. many relatively hyperbolic groups including geometrically finite hyperbolic groups, and fundamental groups of graph manifolds all of whose pieces are hyperbolic.

We prove the first of these by means of expanding endomorphisms. The parade example of an expanding endomorphism is the endomorphism of the integers $n \mapsto 10n$. The facts that this map makes everything larger and that its image is finite index combine to give us decimal notation. Our Cannon’s algorithms for nilpotent groups consist of this sort of decimalization together with cancellation. We are then able to combine these methods with the usual word hyperbolic Cannon’s algorithms to produce the second class of results.

We are also able to prove that many groups do not have Cannon’s algorithms. We have the following criterion: suppose G has two subsets, S_1 and S_2 and that both of these are infinite and the growth of S_2 is exponential. Suppose also that these two sets commute. Then G does not have a Cannon’s algorithm. This allows us to rule out many classes of groups including Baumslag–Solitar groups, braid groups, Thompson’s group, solvegeometry groups and the fundamental groups of most Seifert fibered spaces. In particular, we are able to say exactly which graph manifolds have fundamental groups which have Cannon’s algorithms.

We have discussed Cannon’s algorithms which are carried out by incremental rewriting algorithms and non-incremental rewriting algorithms. They can also be carried out non-deterministically. Given a finite set of length reducing rewriting rules, these solve the word

problem nondeterministically if for each word w , w represents the identity if and only if it can be rewritten to the empty word by some application of these rules. All of these competing versions are closely related to the family of *growing context sensitive languages*. A growing context sensitive grammar is one in which all the productions are strictly length increasing. It is a theorem that a language L is a growing context sensitive language if and only if there is a symbol s and a set of length reducing rewriting rules such that a word w is in L if and only if it can be rewritten to s by some application of these rules. While the family of languages with non-deterministic Cannon's algorithms and the family of growing context sensitive languages may not be exactly the same, our criterion for showing that a group does not have a Cannon's algorithm also seems likely to show that its word problem is not growing context sensitive. Now all automatic groups (and their finitely generated subgroups) have context sensitive word problems [26]. Thus extending this result to the non-deterministic case would show that the class of groups with growing context sensitive word problem is a proper subclass of those with context sensitive word problem.^c

2. Basic Properties

Let us start by justifying the term incremental rewriting algorithm.

Lemma 2

Let $R: \mathbb{A}^* \rightarrow \mathbb{A}^*$ denote reduction by a fixed incremental rewriting algorithm. Then for all $u, v \in \mathbb{A}^*$, $R(uv) = R(R(u)v)$.

Proof—If a substitution can be made in u , the same substitution will be made in uv . Therefore, in exactly the number of steps the algorithm takes to change u into $R(u)$, it changes uv into $R(u)v$. This shows that $R(u)v$ is an intermediate result of running the algorithm on uv . It follows that both must reduce to the same eventual result, i.e. $R(uv) = R(R(u)v)$.

Proposition 3

Let R denote reduction with respect to a Cannon's algorithm $(g, \mathbb{A}, S)$ for G . Let x, y be words in g^* such that $R(x) = R(y)$. Then x and y represent the same element of G .

Proof—If $R(x) = R(y)$ then $R(x)y^{-1} = R(y)y^{-1}$ from which it follows, by Lemma 2, that $R(xy^{-1})$ equals the empty word. But since R comes from a Cannon's algorithm, this implies that x and y represent the same group element.

This means that a Cannon's algorithm always remembers what element of the group it was fed. In a sense this tells us that $R(x)$ is a kind of "canonical form" for $x \in g^*$.

As we shall see in the remarks following Theorem 15, Proposition 3 does not hold for non-incremental Cannon's algorithms. The following proposition shows that the incremental rewriting algorithms form a subclass of the non-incremental ones.

Proposition 4

Given rewriting rules $(\mathbb{A}, S)$ there is a set of rewriting rules $(\mathbb{A}, S')$ such that the non-incremental rewriting algorithm of $(\mathbb{A}, S')$ carries out exactly the same substitutions as the incremental rewriting algorithm of $(\mathbb{A}, S)$.

^cExamples of groups with context sensitive word problem, but not growing context sensitive word problem are given in [20]. In [17] we show that a language is growing context-sensitive if and only if it is the language of a non-deterministic Cannon's algorithm. In addition, we show that the methods of Secs. 6 and 7 extend to these non-deterministic Cannon's algorithms. This has additional language-theoretic consequences.

Proof—Suppose we carry out the non-incremental rewriting algorithm given by $(\mathbb{A}, S)$. In what situation would it make a different substitution to that chosen by the incremental rewriting algorithm? Clearly only when we encounter nested left-hand sides in our word. In that case the non-incremental algorithm chooses the longer word because it starts first, whereas the incremental algorithm chooses the shorter because it ends first. But this means that the incremental rewriting algorithm will never actually invoke the rule with the longer left-hand side. Therefore we can discard from S any rules whose left-hand sides contain another left-hand side ending before the last letter. Call the set of rules we obtain S' . Using these rules both algorithms make exactly the same substitutions.

2.1. Rewriting algorithms and compression

The key result underlying the group theoretic properties of Cannon's algorithms is Theorem 13 which states that if a group has a Cannon's algorithm with respect to one (finite) set of generators, it has one with respect to any other. This follows from a closure property of rewriting systems which we prove in this section.

Consider a group G with finite generating sets g and g' , and suppose there is a Cannon's algorithm $(g, \mathbb{A}, S)$. Each element of $g \in g'$ can be written as a word $w_g \in g^*$. We take these words as the letters of a new generating set g' . Our problem is solved if we can parlay the Cannon's algorithm $(g, \mathbb{A}, S)$ into a Cannon's algorithm whose input letters correspond to words of bounded length in g^* .

Definition 5—Fix $n > 0$ and finite alphabet $\mathbb{A}$. Then $\mathbb{A}^{*n}$ is the set of all nonempty words of $\mathbb{A}^*$ of length at most n . In addition to being elements of $\mathbb{A}^*$, the elements of $\mathbb{A}^{*n}$ also generate the free monoid $(\mathbb{A}^{*n})^*$. The *writing out map*

$$\text{Wr}: (\mathbb{A}^{*n})^* \rightarrow \mathbb{A}^*$$

is the monoid homomorphism induced by the identity on $\mathbb{A}^{*n}$, so called because $\text{Wr}(w)$ writes out each letter of w in $\mathbb{A}^*$.

Note that if $m > n$ then the writing out map for $\mathbb{A}^{*m}$ is the restriction to $(\mathbb{A}^{*n})^*$ of the writing out map for $(\mathbb{A}^{*m})^*$.

In what follows, we use R to denote S reduction of a word in $\mathbb{A}^*$ and R' to denote S' reduction of a word in $(\mathbb{A}^{*n})^*$.

Theorem 6—Let $(\mathbb{A}_0, \mathbb{A}, S)$ be an incremental (or non-incremental) rewriting algorithm. Then for any integer $n > 0$ there exists an incremental (resp. non-incremental) rewriting algorithm $(\mathbb{A}_0^{*n}, \mathbb{A}, S')$ with the following property. For each word $w \in (\mathbb{A}_0^{*n})^*$, $\text{Wr}(R'(w))$ is an intermediate result of S reduction of $\text{Wr}(w)$. $R'(w)$ is the empty word if and only if $R(\text{Wr}(w))$ is the empty word.

This does what is needed: we identify the elements of g' with the words $w_g \in g^*$ representing them. We take $n = \max(\{\ell(w_g) \mid g \in g'\})$. Restricting from $(g^{*n}, \mathbb{A}, S')$ to $(g', \mathbb{A}, S')$ gives a Cannon's algorithm for G with respect to g' .

We will give the proof of Theorem 6 in the incremental case and then indicate the necessary changes for the non-incremental case.

It is clear that simply rewriting S rules into $(A^{**})^*$ does not work. The reason for this is that an S rule may need to decrease A length by n in order to be length-decreasing when written in $(A^{**})^*$. Our remedy for this is three-fold:

1. Use S' rules which result from multiple applications of S rules and therefore delete sufficient A letters to be length-decreasing in A^{**} .
2. Where this is not possible, insert widely spaced letters of $\epsilon \backslash A^{**}$ which make these S' rules length-decreasing. The fact that these letters are sufficiently widely spaced allows rules of the previous type to consume them.
3. Omit certain anchored rules if they fail to delete their left-hand side completely. This causes the S' rules to “stop short” of the corresponding S reduction only in cases where the S reduction did not terminate with the empty word.

The following simplifies some book-keeping.

Lemma 7—Suppose (A_0, A, T) is an incremental rewriting system and that W is the length of the longest left-hand side in T . Then there is an incremental rewriting (A_0, A, S) in which each rule of S is either anchored or has a left-hand side of length W and which carries out the same reductions as (A_0, A, T) .

Proof: Let T^A, T^U be the anchored and unanchored rules of T . For each non-anchored rule $u \rightarrow v$ of T , we take S^U to be the set of rules of the form $au \rightarrow av$ such that $u \rightarrow v \in T^U$, $\ell(a) = W - \ell(u)$ and au contains no T left-hand side which is earlier than u . We take S^A to be the set of rules of the form $a\hat{u} \rightarrow av$ such that $u \rightarrow v \in T^U$, $\ell(a) < W - \ell(u)$ and au contains no T left-hand side which is earlier than u . We take $S = S^U \cup S^A \cup T^A$. It is easy to see that S carries out exactly the same reductions as T .

We will assume that S of Theorem 6 is of this form.

Let us start by visualizing the action of a machine which carries out Cannon’s algorithm. The machine is presented with a word w written on a tape. It views this word through a window of width W . We visualize the window as shrinking when it ends less than W letters from the beginning of w . The window starts at the beginning of w containing a single letter. At each step, it is looking for a left-hand side occupying the entire window. (In the case of an anchored rule, the window may be less than W letters wide.) If it does not see one, it moves one letter to the right or expands by one letter if currently narrower than W letters or terminates if it is at the end of the tape. If it sees a left-hand side it makes the appropriate replacement. (Blanks magically evaporate.) The machine then moves the left edge of the window $W - 1$ letters to the left or shrinks to the appropriate initial segment of w in the case that its left edge was less than $W - 1$ letters from the beginning of w . One may formalize the state of this computation as (w_i, n) where w_i is the current word and $1 \leq n \leq \ell(w_i)$ is the position of the right end of the window.

We now examine the process of *subword reduction* which results from the actions of the machine while the window is confined to a neighborhood of the first left-hand side u in w . (The first S left-hand side in w is the one that ends first and if several end at the same place, it is the longest such.) Consider $w = xUy = xauby$ where u is the first left-hand side in w . We start with the window positioned so that its right edge coincides with the last letter of u . The actions of the machine are determined by aub until one of the following happens: The window encounters a letter of x , or, the window discovers the beginning of the tape because it makes a replacement within $W - 1$ letters of x and x is short, or, the window encounters a letter of y , or the subword y is empty and the machine terminates at the end of b . In the last two cases, we say the machine

goes to the right. This localization of the action of the machine to a neighborhood of the first S left-hand side will allow us to produce S' .

We fix constants

$$\begin{aligned} A_1 &= 2nW + W \\ A_2 &= A_1 + 2n - 2 \\ B_1 &= 2nW \\ B_2 &= B_1 + n - 1 \\ D &= 2n - 1 \\ K &= (W - 1)(2n - 1). \end{aligned}$$

We consider words of the form $U = aub$ where u is the first S left-hand side in U , $\ell(a) \leq A_2$, $\ell(b) \leq B_2$. We allow the machine to proceed with subword reduction on U until it has either made D replacements or has gone to the right. Let V be the result of this subword reduction. Since the machine makes at most D replacements, the left edge of the window moves to the left at most K letters. Accordingly, if $\ell(a) \geq A_1$, the window cannot encounter either letters to the left of U or the beginning of the tape. The rule $U \rightarrow V$ is *acceptable* if $\ell(a) \geq A_1$ and the machine makes D reductions or if $\ell(a) \geq A_1$ and $\ell(b) \geq B_1$. The anchored rule $\hat{U} \rightarrow V$ is *acceptable* if $\ell(a) < A_1$ and the machine makes D reductions or if $\ell(a) < A_1$ and $\ell(b) \geq B_1$ or if $\ell(a) < A_1$, $\ell(b) < B_1$ and the machine deletes U , i.e., V is empty. We take S_2 to be the set of acceptable rules. We take S_1 to be the subset of S_2 rules in which $\ell(a) \leq A_1$ and $\ell(b) \leq B_1$. We use R_i to denote the result of S_i reduction.

Lemma 8—Given $w \in \mathbb{A}_0^*$, for $i = 1, 2$, $R_i(w)$ is an intermediate result of the S reduction of w . $R_i(w)$ is empty if and only if $R(w)$ is empty.

Proof: Suppose that $w = xUy$ where $U = aub$ (or x is empty and $\hat{U} = a\hat{u}b$ if the rule is anchored) is the first S_i left-hand side in w and u is the first S left-hand side in U . We claim that u is the first S left-hand side in w . If the rule is $\hat{U} \rightarrow V$ then u is the first S left-hand side in an initial segment of w and so is the first S left-hand side in w . Suppose, on the other hand, that U is not anchored and that $w = x_1u_1y_1$ where u_1 is earlier than u . Now, u_1 cannot appear in a since u is the earliest S left-hand side in U . Since $U \rightarrow V$ is non-anchored, $\ell(a) \geq A_1 = B_1 + W$. It follows that there are at least B_1 letters following u_1 and ending before the end of b . This implies that u_1 is part of an S_i left-hand side U_1 which is earlier than U , which is a contradiction. Thus the process of S_i reduction takes place by carrying out subword reductions each of which is initiated by the S left-hand side where S would have made its first reduction. Consequently, $R(w)$ is an intermediate result of S reduction.

Now suppose that S makes a reduction at $w = xuy$ which S_i fails to make. This implies that there are fewer than B_1 letters following u , the machine does not make D reductions and does not delete w . This implies that the machine terminates with a non-empty result as required.

Before we can define S' , we need some notation. We define the *reduction point* of a reducible word $w \in \mathbb{A}^*$ to be the last letter of the first S left-hand side in w . We take $\mathbb{B} = \mathbb{A}^{*-}$ and $\mathbb{C} = \mathbb{A}^{*-} \setminus \mathbb{B}$. If $p \in (\mathbb{A}^{*-})^*$ with $\text{Wr}(p) = w$, we say that p is a *spelling* of w . If $\text{Wr}(p)$ is S -reducible, we say that a letter of p is *to the left of the reduction point* if its image in $\text{Wr}(p)$ ends no later than the reduction point in $\text{Wr}(p)$. We say that two letters of p are *separated by at least $K \mathbb{A}$ letters* if their images in $\text{Wr}(p)$ are separated by at least K letters. A spelling of w is *admissible* if each pair of $\mathbb{C}$ letters is separated by at least $K \mathbb{A}$ letters and if $\text{Wr}(p)$ is S -reducible, all $\mathbb{C}$ letters are to the left of the reduction point.

For each $U \rightarrow V \in S_2$, we consider all admissible spellings U' of U . We claim that for each such U' , there is an admissible spelling V' of V so that $U' \rightarrow V'$ is length-decreasing. If V is empty, there is nothing to show. If $U \rightarrow V$ makes $D = 2n - 1$ replacements, then V can be spelled with at least one less $\mathbb{A}$ letter, and this can be done while maintaining the spacing of $\mathbb{C}$ letters. Note that in the subword reduction process, the window moves at most K letters to the left. Accordingly, it encounters at most one $\mathbb{C}$ letter. By removing this letter (if such exists), we ensure that after the subword reduction process, all $\mathbb{C}$ letters are still to the left of the reduction point. Finally, if the machine goes to the right but does not delete $2n - 1$ letters, we may need to introduce an additional $\mathbb{C}$ letter. We do this at the end of V' . We need to check that this preserves admissibility. Now, all $\mathbb{C}$ letters of U' are to the left of the reduction point. But $\ell(b) \geq B_1$ and $U \rightarrow V$ decreases $\mathbb{A}$ length by at most $2n$. We may think of these letters as being removed from the b portion of U . This leaves our new $\mathbb{C}$ letter at least distance K from the previous $\mathbb{C}$ letters, as required.

We call such a rule $U' \rightarrow V'$ an *admissible spelling* of $U \rightarrow V$. We take S' to be the set of admissible spellings of the rules of S_2 . The anchored rules of S' are the admissible spellings of the anchored rules of S_2 .

The following lemma completes the proof of Theorem 6 in the incremental case.

Lemma 9—Suppose $p \in (\mathbb{A}_0^{*n})^*$. Then

- p is an admissible spelling of $\text{Wr}(p)$ and the process of S' reduction preserves admissibility.
- $\text{Wr}(R'(p))$ is an intermediate result of S reduction of $\text{Wr}(p)$. $R'(p)$ is empty if and only if $R(\text{Wr}(p))$ is empty.

Proof: We prove the first by induction on the number of S' reductions. Clearly p is admissible since it has no $\mathbb{C}$ letters. Assume we have reduced it to $p_i = xU'y$ where U' is the first S' side and that this is admissible. We must show that $xV'y$ is admissible. Notice that x obeys the condition on distance between $\mathbb{C}$ letters and y contains no $\mathbb{C}$ letters.

If $U \rightarrow V$ is a rule which introduces a new $\mathbb{C}$ letter, we have already shown that this preserves the condition on the distance between $\mathbb{C}$ letters within V' and does not introduce any new letters at the beginning of V . Consequently, xV' obeys this condition. In this case, the machine acting on $\text{Wr}(xU'y)$ goes to the right, so that the reduction point (if any) in $\text{Wr}(xV'y)$ is in the $\text{Wr}(y)$ portion. Hence $xV'y$ is admissible. Likewise, if V is empty, the reduction point (if any) must lie in $\text{Wr}(y)$ and $xV'y = xV'$ is admissible. Suppose finally, that $U \rightarrow V$ is a rule which makes D substitutions. As we remarked above, the subword reduction process ensures that we can consume any $\mathbb{C}$ letter encountered, ensuring that if $xV'y$ has a reduction point, all $\mathbb{C}$ letters are still to the left of it.

Now consider an admissible word p_i and suppose that this is S' reducible. It follows that $\text{Wr}(p_i)$ is S_2 reducible. Let $\text{Wr}(p_i) = xU_2y$ with U_2 the first S_2 left-hand side. Let U'_2 be the smallest subword of p_i whose image under Wr contains U_2 . The image may contain up to $2n - 2$ extra $\mathbb{A}$ letters at the beginning and $n - 1$ extra $\mathbb{A}$ letters at the end. Thus, it may happen that U'_2 is not an S' left-hand side. However, such a U_2 will contain as a subword a maximal S_1 left-hand side, U_1 . If we then look to U'_1 , the minimal subword of p_i which covers U_1 , this is necessarily an S' left-hand side. U'_1 may be contained in a longer S' left-hand side tU'_1 . The arguments of Lemma 8 now show that this latter occurs at the first S_2 left-hand side of $\text{Wr}(p_i)$ and is a subword reduction starting at the first S left-hand side of $\text{Wr}(p_i)$. We proceed as before.

The proof of Theorem 6 in the non-incremental case requires the following changes. Lemma 7 is easily modified to cover the non-incremental case: we pad rules on the right, so that $u \rightarrow v$ becomes $ub \rightarrow vb$ and end-anchor ub if $\ell(ub) < W$. For a machine carrying out the non-incremental algorithm, the window only shrinks when encountering the end of the tape. Subword reduction is now carried on as before except that this process can now end with when the machine discovers the end of the tape and the definition of the first S left-hand side in w also changes. The first S left-hand side in w is now the one that begins first and if several begin at the same place, it is the longest such. For S rules, this distinction is only relevant for anchored rules since non-anchored left-hand sides all have length W . The definition of S_2 proceeds as before except that we now end-anchor any left-hand side $U = aub$ with $\ell(b) < B_1$.

In the proof of Lemma 8 we must check that the non-incremental algorithm applying S_i applies the subword reduction rule starting at the first S left-hand side. If the S_i left-hand side $U = aub$ is an initial segment of w , then any earlier S left-hand side u_1 must extend past the end of b , for otherwise it would be an earlier S left-hand side in U . This implies $\ell(u) < W$ which implies that u is end-anchored, contradicting the assumption that u_1 extends past u . If U is not an initial segment, then any earlier S left-hand side u_1 must start earlier than a . This implies $\ell(a) \geq A_1$ and we proceed as before.

In the non-incremental case, we define the reduction point of an S -reducible word in $\mathbb{A}^*$ to be the first letter of the first left-hand side. We take S' as before, except that in addition to start-anchored rules, we can also have end-anchored rules. However, with our change of definition of reduction point, if we invoke an S_i rule $U \rightarrow V$ say, in $xUy \rightarrow xVy$, it can now happen that the reduction point appears within the last $W - 1$ letters of V . Consequently, in the definition of S' , if a rule requires the introduction of an additional $\mathbb{C}$ letter, this $\mathbb{C}$ letter may end up to $W - 1 + n - 1 \mathbb{A}$ letters from the end of V' . This still puts it at least $(2nW - e(n - 1) - (W - 1)) > (2n - 1)(W - 1) = K$ letters to the right of the previous reduction point. The proof now proceeds as before.

2.2. Composition of non-incremental rewriting algorithms

Let us introduce the notion of a finite state Dehn machine. As with rewriting algorithms these can be either incremental or non-incremental. (We describe the non-incremental version: to obtain the incremental version, read “ending at the current position” wherever the definition says “starting at the current position.”) Such a machine comes with a finite collection of states, $Q = \{q_i\}$. One of these, q_0 is the start state. For each state $q \in Q$ there is a collection of length reducing replacement rules $S_q = \{u_i \rightarrow v_i\}$. There is also a transition function which chooses a new state depending on the current state and the contents of the subword of length W starting at the current position, where W is an upper bound for the lengths of all the left-hand sides.

Such a machine starts in the start state at the beginning of the input word. In state q , it looks at the next W letters for the longest left-hand side in S_q starting at the current position, and to determine its new state. It then either substitutes and steps W letters to the left, or steps one letter to the right. In either case it switches to the new state. It terminates when it reaches the end of the word with no further replacements possible.

Observe that when a Dehn machine with state terminates it does not necessarily leave behind a word which is free of left-hand sides. While Dehn machines with state are ostensibly more powerful than rewriting algorithms, we show that, by storing the state information in the current word, we can get a rewriting algorithm to “mimic” a Dehn machine. We then use Dehn machines with state to show that non-incremental rewriting algorithms have a nice composition property.

We can extend the concept of *writing out* to include any map $\mathbb{A}^* \rightarrow \mathbb{A}^*$ induced by a map $\mathbb{A}' \rightarrow \mathbb{A}^*$. A machine *stops short* if it terminates at a point when all remaining substitutions would have applied to a final segment of bounded length. One machine *mimics* another if the result of the mimic written out is always a result of the original stopping short.

Proposition 10—Given a non-incremental (or incremental) finite state Dehn machine, there is a non-incremental rewriting algorithm (resp. incremental rewriting algorithm) which mimics it. The mimic terminates with an empty word if and only if the finite state Dehn machine terminates with an empty word *in its start state*.

Proof: We give first a non-strictly length decreasing rewriting algorithm. At the end we sketch how the trick used in the proof of Theorem 6 of introducing widely spaced “multi-letter” letters allows us to give strictly length-decreasing rules. The reason we prefer to give a non-strictly length decreasing algorithm here is that, while the details of making strictly length-decreasing rules are not hard, they would obscure the basically simple idea behind this proof.

Let $\mathbb{A}$ be the working alphabet of our Dehn machine. We make copies of $\mathbb{A}$ in different colors, one corresponding to each state of the machine, and one more in white. The input alphabet, and the copy of $\mathbb{A}$ corresponding to the start state, we color indigo. At any given time during the running of the mimic algorithm an initial segment (possibly empty) of the current word is white. The first colored letter indicates a state of the Dehn machine and its current position, and the remaining letters are all indigo.

Let W be the length of the longest left-hand side of the Dehn machine. We specify the substitutions we wish the mimic to make rather than giving the precise rules. Look W letters to either side of the first colored letter. If a substitution is indicated (according to the state of the first colored letter) we make it, color up to $W - 1$ letters indigo, and one the color of the new state. If no substitution is indicated, the first colored letter is turned white and the next letter is colored with the new state. A special case arises for rules which delete their whole left-hand side and do not lead to the start state. Since there is no suitable letter to color with the new state the mimic instead writes a colored blank. We then have to add a few more rules which take a colored blank followed by a letter and write the same letter in that color.

It is not hard to see that the mimic and the Dehn machine make essentially the same substitutions. When the mimic terminates it is with a word that is white except for its final letter which indicates the termination state of the Dehn machine. If the Dehn machine terminates with an empty word, in a non-input state, the mimic leaves behind a single colored blank.

To make these rules length decreasing we instead look $2W$ letters before and after the first colored letter in the incremental case ($2W$ before and $3W$ after if non-incremental). We run the Dehn machine as a subword reduction. If no substitutions are made, the first colored letter is shifted at least $2W$ letters to the right and two white letters are replaced by one encoding them both. If subword reduction goes to the left, enough substitutions will be made to allow us to remove any “double” letters we find on the way (these ending at least $2W$ original letters apart). If we cannot see $2W$ (resp. $3W$) letters to the right we may have to discard the relevant rule and allow the mimic machine to terminate a little prematurely. This only happens in cases where the machine is unable to make any further substitutions between the current point and the end of the word.

Let $(\mathbb{A}, S)$ and $(\mathbb{A}, S')$ be non-incremental rewriting algorithms with reduction maps P and P' respectively. Ideally there would then be a non-incremental rewriting algorithm with reduction map $P' \circ P$. Unfortunately this does not appear quite to be the case. We have to allow the

resulting algorithm to give its answer in some “compression alphabet” $\mathbb{A}^*$, and we may have to allow it to stop short of reaching its answer. We do not really mind the compression alphabet, but having a machine stop short is a problem: it gets in the way of doing any further composition.

Reluctantly, we must add a further “flavor” of non-incremental rewriting algorithm to our collection. An *nearly strict non-incremental rewriting algorithm* is one which may include some length preserving *ending rules*: these are end anchored rules such that the resulting algorithm has the property that one of these will apply only when the word is reduced with respect to all the strictly length decreasing rules, and afterwards the word will be fully reduced. We shall not consider here the question of how to determine, in general, whether a given set of rules has this property. What is hopefully clear is that if, in the proof of Theorem 6, we put back the deleted rules, we obtain a nearly strict non-incremental rewriting algorithm. Modulo writing out, the resulting algorithm achieves the reduction map of the original algorithm. Furthermore it makes no difference to the proof if the original algorithm is itself nearly strict.

Lemma 11—Let $(\mathbb{A}_0, \mathbb{A}, S)$ be a nearly strict non-incremental rewriting algorithm. Then for any integer $n > 0$ there exists a nearly strict non-incremental rewriting algorithm $(\mathbb{A}_0^{*n}, \mathbb{A}, S')$ with the following property. For each word $w \in (\mathbb{A}_0^{*n})^*$, the reduction of w , with respect to $(\mathbb{A}_0^{*n}, \mathbb{A}, S')$, written out is the reduction of w written out with respect to $(\mathbb{A}_0, \mathbb{A}, S)$.

Similarly, when we construct a mimic for a non-incremental Dehn machine with state, we can avoid stopping prematurely by allowing ending rules for the resulting machine. We can also allow a Dehn machine with state to have ending rules. These are length preserving rules which put it into a *terminal state*, a state without rules which the machine cannot leave. Such a machine can also be mimicked by a nearly strict non-incremental rewriting algorithm, the proof being virtually unchanged.

We shall show that it is possible to compose nearly strict non-incremental rewriting algorithms. We can always recover a genuine non-incremental rewriting algorithm which might stop short, by discarding the length preserving rules.

Proposition 12—Let $(\mathbb{A}, S)$ and $(\mathbb{A}, S')$ be nearly strict non-incremental rewriting algorithms with reduction maps P and P' respectively. There is a nearly strict non-incremental rewriting algorithm $(\mathbb{A}, \mathbb{A}', T)$ which mimics the process of first applying $(\mathbb{A}, S)$ and then applying $(\mathbb{A}, S')$. The reduction map of $(\mathbb{A}, \mathbb{A}', T)$ written out is the composition $P' \circ P$.

Proof: This process can be carried out by a finite state Dehn machine. In its initial state it applies the rules in S . Once no more rules apply and it approaches the end of the word, it switches to a second state. In this state it simply compresses a little bit until it arrives at the start of the word again. Then it switches into a third state where it uses the rules in S' modified, as in Lemma 11, for compressed input.

What if S includes ending rules? Without loss of generality, S -left-hand sides are either $W > 3$ letters long, or anchored at both ends. Ending rules of length W can be combined with compression. Rules anchored at both ends can be modified so as to complete the entire reduction $(P' \circ P)$ at a single step.

With S as above, our machine can recognize the end of a reduced word by finding any word of $W - 1$ letters which is anchored at the end but not the start. (Shorter entire words being already dealt with.) It can then start backtracking and compressing.

When the modified S' has ending rules, these become ending rules for the finite state machine. Finally, we transform the resulting Dehn machine with state into a nearly strict non-incremental rewriting algorithm.

2.3. Group theoretic consequences

From Theorem 6 and the discussion at the start of this section we have the following result.

Theorem 13—Let G be a group with finite semi-group generating sets g and g' . Then G has a Cannon's algorithm with respect to g if and only if it has one with respect to g' .

Theorem 14—Let G be a group and let H be a finitely generated subgroup of G . If G has a Cannon's algorithm, H has one too.

Proof: Choose a set of generators for G which includes generators for H . With respect to these generators, a Cannon's algorithm for G is also one for H .

Theorem 15—Let G be a group and let H be a finite index subgroup of G . If H has a Cannon's algorithm, G also has one.

Proof: Fix a transversal T for $[H: G]$ from which we omit the representative of the identity coset. Fix a finite generating set g for G containing T . Each word $g_1 g_2 g_3$ with $g_i \in G$ is equal in G to a word of the form $[h][t]$, for some $h \in H$ and $t \in T$, where the brackets indicate that each letter may be omitted. If $g_1 g_2 \in g^*$ evaluates to an element of H , it can be written as the 0 or 1-letter word $[h]$, again for some $h \in H$. As g_1, g_2, g_3 vary in g we obtain finitely many elements $h \in H$. Let $\mathcal{H}$ be a finite generating set for H containing all non-identity elements obtained in this way and also, all of $g \cap H$.

The above equalities give rules R of the form $g_1 g_2 g_3 \mapsto ht$, etc. We omit any rules with $g_1 \in \mathcal{H}$. The incremental rewriting algorithm $(g, g \cup \mathcal{H}, R)$ turns a word in g^* into a word in $\mathcal{H}^*$ followed by at most two letters from g by pushing a coset representative along the word. If an input word to this algorithm represents an element of H , the reduced word will be in $\mathcal{H}^*$.

Let $(\mathcal{H}, \mathbb{A}, S)$ be a Cannon's algorithm for H . We claim that $(g, g \cup \mathcal{H} \cup \mathbb{A}, R \cup S)$ is a Cannon's algorithm for G . The R rules translate the word into a word in $\mathcal{H}$ followed by a couple of letters keeping track of the coset. Then the S rules chase along behind applying H 's Cannon's algorithm to the word in $\mathcal{H}$. The effect is exactly as if we applied $(g, g \cup \mathcal{H}, R)$ first, followed by applying $(\mathcal{H}, \mathbb{A}, S)$ to the $\mathcal{H}^*$ part of the result. If an input word represents the identity in G , the first step produces a representation of the identity in $\mathcal{H}^*$ and the second deletes it. An input word which does not represent the identity will reduce, either to some word containing letters in $g - \mathcal{H}$, if it does not evaluate into H , or otherwise to a non-empty word in $(\mathcal{H} \cup \mathbb{A})^*$.

The previous theorems hold both for both Cannon's algorithms and non-incremental Cannon's algorithms. The last of these suggests a way to construct a Cannon's algorithm using the non-incremental rewriting algorithm which does not satisfy Proposition 3. Consider the case of H finite index in G . It is not hard to parlay a Cannon's algorithm for H into a non-incremental rewriting algorithm which solves the word problem in G but destroys information in the case where the word is not in the identity coset. Here is what it does: given a word w , it first transforms this into a word of the form ht where h is a word in the generators for H (possibly the empty word) and t is an element of the transversal, and is empty if and only if it represents the identity coset. If t is empty, we now proceed to reduce h according to the Cannon's algorithm for H . On the other hand, if t is not empty, armed with right anchored rules, we can proceed with the wanton destruction of the information in h .

Proposition 16—There is a non-incremental rewriting algorithm which is not mimicked by any incremental rewriting algorithm.

Theorem 17—If G and H both have Cannon's algorithms, then so does their free product $G * H$.

Proof: We suppose that G_0 and G_1 are groups with Cannon's algorithms $(g_0, \mathbb{A}_0, S_0)$ and $(g_1, \mathbb{A}_1, S_1)$ respectively and that the alphabets for these are disjoint. Let

$$\begin{aligned} T_0 &= \{au \rightarrow av\widehat{u} \rightarrow v \in S_0, a \in \mathbb{A}_1\} \\ T_1 &= \{au \rightarrow av\widehat{u} \rightarrow v \in S_1, a \in \mathbb{A}_0\} \\ S &= S_0 \cup T_0 \cup S_1 \cup T_1 \\ \mathcal{G} &= \mathcal{G}_0 \cup \mathcal{G}_1 \\ \mathbb{A} &= \mathbb{A}_0 \cup \mathbb{A}_1. \end{aligned}$$

We claim that $(g, \mathbb{A}, S)$ is a Cannon's algorithm for $G_0 * G_1$.

To see this, consider a word $x_0 \dots x_n$ consisting of alternating non-empty words from the alphabets g_0 and g_1 . For simplicity, we will assume that we have numbered the two groups so that $x_i \in \mathcal{G}_{i(\bmod 2)}^*$. We claim that as long as no x_i evaluates to the identity, $R(x_0 \dots x_n) = R_0(x_0) \dots R_n(x_n)$. (Here we are using R to denote reduction with respect to S and R_i to denote reduction with respect to $S_{i(\bmod 2)}$). Likewise, we will refer to $S_{i(\bmod 2)}$ as S_i and $T_{i(\bmod 2)}$ as T_i .)

This claim is true when $n = 0$, for then only the rules of S_0 apply. Suppose now that this claim holds for $n = k$. We wish to establish it for the case $n = k + 1$. By induction an intermediate result of the reduction of $x_0 \dots x_{k+1}$ is $R_0(x_0) \dots R_k(x_k)x_{k+1}$ and the portion before x_{k+1} is fully reduced. Further, the assumption that no x_i evaluates to the identity implies that $R_k(x_k)$ is non-empty. Accordingly any further reductions are made either by a non-anchored rule of S_{k+1} applying entirely inside x_{k+1} or by a rule of T_{k+1} applying at the last letter of $R_k(x_k)$ and the beginning of x_{k+1} . Any rule of T_{k+1} changes only the letters of x_{k+1} and performs exactly as an anchored rule of S_{k+1} would have done had x_{k+1} been the beginning of a word. These combine to produce $R_0(x_0) \dots R_k(x_k)R_{k+1}(x_{k+1})$ as required.

In particular if no x_i represents the identity, then $x_0 \dots x_n$ does not represent the identity and does not reduce to the empty word.

Now consider the case in which some x_i represents the identity. We take x_i to be the earliest such. The process of reducing the word $w = x_0 \dots x_n$ produces $R_0(x_0) \dots R_{i-1}(x_{i-1})x_i \dots x_n$ as an intermediate result. As before, S_i and T_i conspire to reduce x_i as S_i would have done had x_i stood alone. This produces $R_0(x_0) \dots R_{i-1}(x_{i-1})x_{i+1} \dots x_n$. But this is an intermediate result of reducing $w' = x_0 \dots x_{i-1}x_{i+1} \dots x_n$. Furthermore, w' represents the identity if and only if w represents the identity. But the free product length of w' is two less than the free product length of w . Thus we may assume inductively that w' reduces to the empty word if and only if it represented the identity and we conclude the same about w .

Since this induction reduces free product length by two, it remains to check two base cases. One is when the free product length of w is 0, and here there is nothing to check. The second is when the free product length is 1. This is just application of the Cannon's algorithm in one of the factor groups.

We do not know how to prove this for non-incremental Cannon's algorithms. This raises the following

Question 18—Are there groups with non-incremental Cannon's algorithms which do not have Cannon's algorithms?

With an additional assumption on the Cannon's algorithms of the factors, we will extend this to free products with amalgamation over finite subgroups in Corollary 41. This raises the following

Question 19—Are groups with Cannon's algorithm closed under free product with amalgamation over finite subgroups?

3. Groups with Expanding Endomorphism

Let G be a finitely generated group with finite set of semi-group generators $\mathcal{g}$. Let $\ell_{\mathcal{g}}$ denote the word metric on G with respect to $\mathcal{g}$. We say that a homomorphism $\phi: G \rightarrow G$ is an *expanding endomorphism* if $\phi(G)$ is a finite index subgroup of G and there exists a constant $M > 1$ such that $\ell_{\mathcal{g}}(\phi(g)) \geq M \ell_{\mathcal{g}}(g)$ for all $g \in G$. Observe that by taking a suitable power of ϕ we may make M as large as we wish. By taking a finite set of coset representatives for $\phi(G) \backslash G$ we see that there is a constant K such that for all $g \in G$, the distance from g to $\phi(G)$ is at most K . We say that $\phi(G)$ is K -dense in G .

Note that an expanding map ϕ is necessarily injective. For if $g \neq g'$ then $\ell(gg'^{-1}) \geq 1$, so $\ell(\phi(gg'^{-1})) \geq M$. This implies, for example, that if $g \notin \phi(G)$ then $\phi(g) \notin \phi^2(G)$.

Let $\mathbb{A}$ be the finite alphabet $\mathcal{g} \cup \{t, t^{-1}\}$, where t and t^{-1} are letters not in $\mathcal{g}$. We say that a word w in $\mathbb{A}$ is *balanced* (with respect to t) if w has the same number of t 's as t^{-1} 's, and further, every initial segment of w has at least as many t 's as t^{-1} 's. Each balanced word w in $\mathbb{A}$ represents an element of G : we define the element represented by $tw t^{-1}$ to be ϕ applied to the element represented by w .

We will show that if ϕ is chosen so that both M and K are sufficiently large, the following rules give a Cannon's algorithm for G . In these rules: g denotes a word in $\mathcal{g}^*$, and g' and g'' denote geodesic words in $\mathcal{g}^*$ such that $g = \phi(g')g''$, and $\ell(g'')$ equals the distance from g to $\phi(G)$.

1. Replace any non-geodesic word g of length $\ell(g) \leq 2K$ by an equivalent geodesic word.
2. If g is geodesic, with $\ell(g) = 2K$, replace g by $tg't^{-1}g''$, and replace $t^{-1}g$ by $g't^{-1}g''$.
3. If g is geodesic, with $\ell(g) \leq 2K$ and $\ell(g'') = 0$ (i.e. $g \in \phi(G)$), replace $t^{-1}g$ by $g't^{-1}$.
4. Replace tt^{-1} by the empty word.

These rules clearly map balanced words to balanced words, and do not change the element of G represented. It is clear that Rules 1, 3 and 4 are strictly length decreasing. For Rule 2 to reduce length we need $\ell(g') + \ell(g'') + 2 < \ell(g)$. We have $\ell(g) = 2K$, $\ell(g'') \leq K$, and

$\ell(g') \leq \frac{1}{M}(\ell(g) + \ell(g''))$. It follows that Rule 2 will be length decreasing if $3/M + 2/K < 1$.

Lemma 20

Let $G, \mathbb{A} = \mathcal{g} \cup \{t, t^{-1}\}$, M and K be as above. Let w be the reduction of a word in $\mathcal{g}^*$ with respect to Rules 1–4. Then w has the form $t^n g_n t^{-1} \dots t^{-1} g_1 t^{-1} g_0$, or just g_0 ($n = 0$), such that:

1. each g_i is a geodesic word in $\mathcal{g}$ of length less than $2K$;
2. each g_i , for $i < n$, is either in $G - \phi(G)$ or it is empty;
3. if $n > 0$, g_n is not empty.

Proof—We show first that all t 's appear at the start of w . Initially this is vacuously true. The only rule whose application could make this untrue is 2 since it is the only rule which creates t 's. But Rule 2 is only applied at the start of the word, or when the immediately preceding letter is t , for otherwise one of Rules 1–3 would apply at least one letter to the left.

Rule 1 ensures that each g_i is geodesic, while Rule 2 ensures that the length of each g_i is less than $2K$. Rule 3 ensures that each g_i , for $i < n$, is either in $G - \phi(G)$ or it is empty. Rule 4 ensures that g_n is not the empty word if $n > 0$.

Theorem 21

Rules 1–4 reduce each word in $\mathcal{g}$ to the empty word if and only if that word represents the identity element of G .

Proof—Let $g = t^n g_n t^{-1} \dots t^{-1} g_1 t^{-1} g_0$ be the reduction of a word in $\mathcal{g}$ representing the identity in G . Let i be the least integer such that g_i is non-trivial. Then if $i < n$, 2 in Lemma 20 implies that g belongs to a non-1 coset of $\phi^{i+1}(G)$. Therefore g_i is trivial for $i < n$.

Hence g_n represents the identity in G . By 1, g_n is geodesic and therefore trivial. By 3, $n = 0$ and so g itself is trivial. The converse is clear.

The process we have just described is essentially that of writing the decimal expansion of a number. Indeed, if you apply this to the sum of 572 1's, $1 + \dots + 1$, using the endomorphism $n \mapsto 10n$ you will get $t^2 5 t^{-1} 7 t^{-1} 2$. This is nothing but the decimal 572 with t 's performing the function of place notation. Unfortunately, our decimal expansions can be a bit perverse. In addition to the numerals for the numbers 0 through 9, we also have numerals for the numbers -1 through -9 . Let us give these the numerals $\hat{1}$ through $\hat{9}$. If you count up to 1, 000, 000 and then count back down to 1, you will wind up writing 1 as $t^6 1 t^{-1} \hat{9} t^{-1} \hat{9} t^{-1} \hat{9} t^{-1} \hat{9} t^{-1} \hat{9} t^{-1} \hat{9}$, i.e., as 1999999. Evidently, we can write an arbitrarily long word for the number 1.

We say that a Cannon's algorithm is *finite to one* if, as x varies over all words representing a fixed element of G , $R(x)$ takes only finitely many values.

Remark 22

There are Cannon's algorithms which are not finite to one off the identity.

For the purposes of Sec. 5 we would like to modify our Cannon's algorithm to avoid this behavior.

Given a reduced word $w = t^n g_n t^{-1} g_{n-1} \dots g_1 t^{-1} g_0$, we call n the *height* of w . Choose a positive integer N such that $(M/3)^N > K$. We add the following additional rules to our system.

(5) If w is a reduced word, as above, with height at most N , such that $\ell_{\mathcal{g}}(w) < \frac{1}{2} \ell(g_0)$, replace w by an equivalent geodesic word in $\mathcal{g}^*$.

Since there are only finitely many reduced words of height at most N , this introduces only finitely many rules.

Lemma 23—Let w be the reduction of a word in $\mathcal{g}^*$ with respect to rules 1–5. If the height of w is n , and w does not represent the identity, then $\ell_{\mathcal{g}}(w) \geq (M/3)^n$.

Proof: For height $n = 0$ the lemma is clear. For $n > 0$ we can write $w = t w' t^{-1} g_0$, where w' is reduced, of height $n - 1$, and not the identity, and g_0 is geodesic. For $n \leq N$, Rule 5 ensures that

$\ell_g(w) \geq \frac{1}{2}\ell(g_0)$. It follows that $3\ell_g(w) \geq \ell_g(w) + \ell(g_0) \geq \ell_g(tw't^{-1}) \geq M\ell_g(w')$. By induction the lemma holds for all $n \leq N$.

For $n > N$, writing w as before, $\ell_g(w) \geq M\ell_g(w') - \ell(g_0)$. By induction, $\ell_g(w') \geq (M/3)^{n-1}$. Also $\ell(g_0) \leq 2K$ which, by our choice of N , is less than $2(M/3)^{n-1}$. Therefore $\ell_g(w) \geq (M-2)(M/3)^{n-1} \geq (M/3)^n$ since $M > 3$.

Corollary 24—If G admits an expanding endomorphism then G is virtually nilpotent.

Proof: Each element $g \in G$ can be represented by a word w whose length is bounded by $k \ln(\ell_g(g) + 1)$, for some $k > 0$. Since there are only polynomially many such words, G has polynomial growth and hence [14] is virtually nilpotent.

It is apparently unknown whether all torsion free nilpotent groups have expanding endomorphisms. However, we will see in the next section that they all have Cannon's algorithms.

Theorem 25—If G has an expanding endomorphism, then G has a finite to one Cannon's algorithm.

Proof: As in Corollary 24, the length $\ell_g(g)$ of an element $g \in G$ gives a bound for the maximum length of any reduced normal form representing g . Therefore there are at most finitely many possible reduced normal forms for each element.

Remark 26—The results of this section remain valid under the weaker hypothesis that G has a finite index subgroup H which admits an expanding endomorphism ϕ with respect to ℓ_g . The only change that needs to be made is to replace $\phi(G)$ with $\phi(H)$ throughout.

This has the following corollary which we will need in our work on geometrically finite groups.

Corollary 27—Let G be finitely generated and suppose that G has a finite index subgroup which has an expanding endomorphism. Let $\mathcal{g}$ be a set of semi-group generators for G . Then for any $N > 0$ there exists a Cannon's algorithm as above, with working alphabet $\mathbb{A} = \mathcal{g} \cup \{t, t^{-1}\}$, such that any normal form word w with $\ell_g(w) < N$ is a geodesic word in $\mathcal{g}^*$. In particular, this holds when G is finitely generated and virtually abelian.

Proof: Let H be a finite index subgroup with expanding endomorphism. (In the virtually abelian case, H is a finite index free abelian subgroup.) Raising to a sufficient power furnishes us with an expanding endomorphism of H , with expansion factor M such that $M/3 > N$. By Lemma 23, any normal form word w with $\ell_g(w) < N$ has height 0.

We will call such a Cannon's algorithm *N-geodesic*.

We will say that a rule $u \rightarrow v$ is a local geodesic rule if both u and v are words in the group generators and v is a geodesic. We will say that an *N-geodesic* Cannon's algorithm $(\mathcal{g}, \mathbb{A}, S)$ is *N-tight* if $(\mathcal{g}, \mathbb{A}, S \cup R)$ is also an *N-geodesic* Cannon's algorithm whenever R is a finite set of local geodesic rules and the left-hand sides of S and R are disjoint.

We record here the following observation.

Proposition 28—The *N-geodesic* Cannon's algorithms of Corollary 27 are *N-tight*.

Proof: In this case, the rules of S determine that any sufficiently long geodesic g is replaced with a word $tg't^{-1}g''$ where g and g' are shorter geodesics. On the other hand, S replaces any non-geodesic shorter than this with a geodesic. In particular, no rule of R is ever applied.

4. Nilpotent Groups

In this section we shall show that every finitely generated, torsion free nilpotent group embeds in a group which has an expanding endomorphism. It follows from Theorem 25 and Theorem 14 that every torsion free nilpotent group has a Cannon's algorithm. Since every finitely generated nilpotent group is virtually torsion free, it follows by Theorem 15 that every finitely generated virtually nilpotent group has a Cannon's algorithm.

We start with the group of $n \times n$ upper triangular matrices with 1's on the diagonal. Those with integer entries we denote by $U_n(\mathbb{Z})$, those with real entries we denote by $U_n(\mathbb{R})$.

For each $\mu \in \mathbb{R}$ define $f_\mu: U_n(\mathbb{R}) \rightarrow U_n(\mathbb{R})$ as follows. If $x = (x_{ij}) \in U_n(\mathbb{R})$, set

$$(f_\mu(x))_{ij} = \mu^{j-i} x_{ij}.$$

It is not hard to see that f_μ is a homomorphism and that if $\mu \in \mathbb{Z}$ then $f_\mu: U_n(\mathbb{Z}) \rightarrow U_n(\mathbb{Z})$.

Let us fix a generating set g for $U_n(\mathbb{Z})$ and endow $U_n(\mathbb{R})$ with a left invariant metric.

Lemma 29

The action of $U_n(\mathbb{Z})$ on $U_n(\mathbb{R})$ is co-compact by isometries and fixed point free.

Proof—It is clear that this action is fixed point free and by isometries. We need to see that each $x = (x_{ij}) \in U_n(\mathbb{R})$, is a bounded distance away from some $z = (z_{ij}) \in U_n(\mathbb{Z})$. For $p = 1, \dots, n-1$, let $m_p(x_1, \dots, x_{n-p})$ be the upper triangular matrix with 1's on the diagonal, $x_1, \dots, x_{n-p}$ located distance p above the diagonal, and 0's everywhere else. Notice that multiplying $x = (x_{ij})$ by such an m_p leaves unchanged the entries of x below the p th off-diagonal and adds $x_1, \dots, x_{n-p}$ to the entries of x on the p th off-diagonal. Consequently, we can choose $m_1, m_2, \dots, m_{n-1}$ each with entries between 0 and 1 so that $z = xm_1m_2 \dots m_{n-1} \in U_n(\mathbb{Z})$. Since the entries of each m_i are bounded in size, so is their product. Hence z is a bounded distance away from x as required.

Consequently,

Lemma 30

There is $\lambda = \lambda_g$ so that the embedding of the Cayley graph $\Gamma_g(U_n(\mathbb{Z}))$ into $U_n(\mathbb{R})$ is a $(\lambda, 0)$ -quasi-isometry.

Proof—It is a standard result that a co-compact discrete isometric action on a geodesic metric space induces a (λ, ε) -quasi-isometry. It is not hard to see that in the case of a fixed point free action, we may take $\varepsilon = 0$.

Lemma 31

For $\mu > 1$, the map f_μ is a μ -expanding endomorphism on $U_n(\mathbb{R})$. That is, for $x, y \in U_n(\mathbb{R})$, $d(f_\mu(x), f_\mu(y)) \geq \mu d(x, y)$.

Proof—It suffices to show that f_μ is everywhere infinitesimally μ -expanding. For $X \in U_n(\mathbb{R})$, a tangent vector at X is given by $\frac{d}{dt}(X+At)$ where $A = (a_{ij})$ with $a_{ij} = 0$ for all $j \leq i$. Without loss of generality we may assume $\|\frac{d}{dt}(I+At)\| = (\sum a_{ij}^2)^{1/2} =: n(A)$. Then by left invariance and linearity of matrix multiplication

$$\|\frac{d}{dt}(X+At)\| = n(X^{-1}A).$$

Clearly f_μ extends linearly to all upper-triangular matrices and we have

$$\begin{aligned} \|\frac{d}{dt}f_\mu(X+At)\| &= \|\frac{d}{dt}(f_\mu(X) + f_\mu(A)t)\| \\ &= n(f_\mu(X^{-1}A)) \geq \mu n(X^{-1}A). \end{aligned}$$

Lemma 32

For $\mu \in \mathbb{Z}$, $f_\mu(U_n(\mathbb{Z}))$ is finite index in $U_n(\mathbb{Z})$.

Proof—The proof is the same as the proof that $U_n(\mathbb{Z})$ is co-compact in $U_n(\mathbb{R})$.

Consequently,

Lemma 33

If $\mu > \lambda^2$, and $\mu \in \mathbb{Z}$ then f_μ is an expanding endomorphism of $U_n(\mathbb{Z})$.

Hence, by Theorem 25,

Lemma 34

$U_n(\mathbb{Z})$ has a finite to one Cannon's algorithm.

Now it is a theorem (see [24, Chap. 5]) that

Theorem 35

If G is a finitely generated, torsion free nilpotent group then G embeds in $U_n(\mathbb{Z})$, for some $n > 0$.

Hence, by Theorems 14 and 15,

Theorem 36

If G is finitely generated and virtually nilpotent, then G has a Cannon's algorithm.

5. Relatively Hyperbolic Groups

In this section we prove a theorem concerning Cannon's algorithms for (strongly) relatively hyperbolic groups. We first proved this in the context of geometrically finite hyperbolic groups and these are the paradigm example of relatively hyperbolic groups. The statement and proof here are close parallels of the geometrically finite case.

There are multiple equivalent definitions of what it means for a group to be (strongly) hyperbolic relative to a collection of subgroups $\{P_1, \dots, P_k\}$. These are equivalent to Farb's

[13] definition of relative hyperbolicity together with his bounded coset penetration property. Usage of the term relatively hyperbolic varies slightly in that it is often possible to drop the requirement that the subgroups be finitely generated. In our usage these will all be finitely generated.

The key geometric result is the relation the geodesics and horoballs of the negatively curved space to the geodesics of subspace upon which the group acts co-compactly. This is Lemma 42 here, the Morse lemma [12, Proposition 8.28].

Theorem 37

Suppose that G is hyperbolic relative to $\mathcal{P} = \{P_1, \dots, P_k\}$. Suppose also that for each i , $1 \leq i \leq k$ and any N , P_i has a Cannon's algorithm which is N -tight. Then G has a Cannon's algorithm. This Cannon's algorithm consists of local geodesic rules together with Cannon's algorithms for the P_i .

Corollary 38

If G is a geometrically finite hyperbolic group, then G has a Cannon's algorithm.

Corollary 39

If M is a graph manifold each of whose pieces is hyperbolic then $\pi_1(M)$ has a Cannon's algorithm.

Corollary 40

Suppose that M is a finite volume negatively curved manifold with curvature bounded below and bounded away from zero. Then $\pi_1(M)$ has a Cannon's algorithm.

Corollary 41

Suppose that A and B are groups with N -tight Cannon's algorithms and that C is a finite group which includes as a subgroup of each of these. Then $A *_C B$ has a Cannon's algorithm.

Corollaries 38 and 39 follow directly from Theorem 37 since the groups in question are hyperbolic relative to abelian (or virtually abelian) groups. Corollary 41 follows since the amalgam is hyperbolic relative to its factors. In the case of Corollary 40, the groups are hyperbolic relative to nilpotent groups [13]. Nilpotent groups have Cannon's algorithms by Theorem 36, but there is no guarantee that these are N -tight for arbitrary N . It is only in the perhaps larger group of upper triangular matrices where this is guaranteed. However, once we have proved Theorem 37, we will see how to proceed here.

We suppose that G is hyperbolic relative to a finite collection of subgroups $\{P_1, \dots, P_k\}$. The parabolic subgroups of G are the G -conjugates of $\{P_1, \dots, P_k\}$. We take $\mathcal{P}$ to be the set parabolic subgroups. The following are well known properties of relatively hyperbolic groups. See, for example, [3,²³] and [12].

Review 42

1. G acts discretely by isometries on a δ -hyperbolic space H .
2. This action induces an action on the boundary ∂H .
3. There is a G equivariant family of horoballs $\{B_P \mid P \in \mathcal{P}\}$.
4. For each $P \in \mathcal{P}$ we take S_P to be ∂B_P . P acts co-compactly on S_P .
5. G acts co-compactly on $X = H \setminus (\cup_{P \in \mathcal{P}} B_P)$.

6. Each horoball B_P is quasiconvex. Consequently, there is a retraction $r_P: X \rightarrow S_P$ which is inherited from the hyperbolic retraction of H onto S_P . (We will also refer to this retraction as r_S where S is the boundary of P .)
7. For points sufficiently distant from S_P , the retraction r_P shrinks X distance by a super-linear factor. That is to say, there is a function $s(\cdot)$ with the property that for any linear function $y(x) = mx + b$, there is x_0 such that for $x > x_0$, $s(x) > y(x)$ and there is d_0 so that if $d = \min(d_X(p, S_P), d_X(q, S_P)) > d_0$ then

$$d_X(r_P(p), r_P(q)) < \frac{d_X(p, q)}{s(d)}.$$

8. There is ζ with the following property. Suppose that S_0 and S_1 are disjoint horospheres, i.e., the boundaries of disjoint horoballs in $\mathcal{P}$. Suppose that γ and γ' are H geodesics that start in S_0 and end in S_1 and that x and x' are the last points of γ and γ' in S_0 . Then $d_X(x, x') \leq \zeta$.
9. There is ζ so that if S_0 and S_1 are disjoint horospheres, then $r_{S_0}(S_1)$ has d_X diameter bounded by ζ .
10. Given ζ there is ε with the following property. Suppose S is the boundary of horoball B . If γ is an H geodesic that starts and ends on S then the only portion of γ lying in the ζ neighborhood of $H \setminus B$ are an initial and terminal segment of γ , each of length at most ε .

We need the following lemma which is [12, Proposition 8.28].

Lemma 43

There is δ depending only on λ and ε with the following property. Suppose that w is a (λ, ε) quasigeodesic in X and γ is an H geodesic with the same endpoints. Suppose that $\Sigma = \Sigma(\gamma)$ is the union of γ and the horospheres that it meets. Then w lies in a δ neighborhood of Σ .

Given an H geodesic, γ , it meets a finite (possibly empty) collection of horoballs, $B_1, \dots, B_k$. Replace each portion $\gamma \cap B_i$ with an X geodesic, σ_i to produce the X paths

$$\sigma = \gamma_0 \sigma_1 \gamma_1 \dots \sigma_n \gamma_n.$$

We refer to a path formed in this way as a *rough geodesic*.

Lemma 44

There is a λ such that every rough geodesic is an $X(\lambda, 0)$ -quasigeodesic.

Proof—First we show that there are λ and ε so that every rough geodesic is a (λ, ε) -quasigeodesic. Suppose that γ is an H geodesic and $\sigma = \gamma_0 \sigma_1 \gamma_1 \dots \sigma_n \gamma_n$ is a corresponding rough geodesic. Suppose that σ' is a corresponding X geodesic. By Lemma 43, this lies in a δ neighborhood of $\Sigma = \Sigma(\gamma)$. Let us decompose σ' as $\sigma' = \gamma'_0 \sigma'_1 \gamma'_1 \dots \sigma'_n \gamma'_n$ where σ'_i is the portion of σ' which lies within δ of the horosphere for σ_i , but not within δ of γ . Some of these may be empty. However, it follows that for each i , γ_i and γ'_i lie within 2δ of each other. Since each of these is geodesic, the difference in their lengths is bounded. Similarly, for each i , the endpoints of σ_i and σ'_i are close to each other, thus bounding the difference in their lengths. Accordingly, the difference in lengths along σ and σ' arise only from these breakpoints each of which contributes only a bounded difference. Since there is a minimum distance between

horospheres, these breakpoints are bounded away from each other. This makes σ a (λ, ε) -quasigeodesic.

We now claim that with appropriate adjustment to λ we can choose $\varepsilon = 0$. First note that by changing λ if necessary, we can ensure that any failure to be a $(\lambda, 0)$ -quasigeodesic occurs “in the small”, i.e., for short subsegments. Since the paths in question lie in a graph, any subpath which is not a loop has its ends at least distance 1 from each other. Thus, the only circumstance under which rough geodesics would fail to be $(\lambda, 0)$ -quasigeodesics for some λ would be if a rough geodesic could contain a closed subpath. Any such loop would have to contain the juncture of a hyperbolic and a horospherical segment. But these meet only at their common endpoint. The result follows.

We record here two general properties of δ -hyperbolic spaces. (Here we use the parameterized version of δ -hyperbolicity.)

Proposition 45

Given $\delta' > \delta$, there are (λ, ε) with the following property. Suppose γ is a piecewise geodesic. Suppose that each segment of γ has length at least $\delta' + 1$, and that at each bend, both segments depart a δ neighborhood of each other after travelling at most distance δ' from that bend. Then γ is a (λ, ε) -quasigeodesic.

Proposition 46

Suppose that $(\lambda_1, \varepsilon_1)$ and $(\lambda_2, \varepsilon_2)$ are given. Then there is $(\lambda_3, \varepsilon_3)$ with the following property. If σ is a $(\lambda_1, \varepsilon_1)$ -quasigeodesic and τ is formed from σ by replacing disjoint subpaths with $(\lambda_2, \varepsilon_2)$ -quasigeodesics, then τ is a $(\lambda_3, \varepsilon_3)$ -quasigeodesic.

Suppose G is hyperbolic relative to $P_1, \dots, P_k$. We would like to find a generating set $\mathcal{g}$ in which $P_1, \dots, P_k$ are convex in the Cayley graph of G . Given a set of generators $\mathcal{g}'$ for G and $K > 0$, set

$$\mathcal{A}_i(K) = \{g \in P_i \mid d_X(1, g) \leq K\}$$

and

$$\mathcal{G} = \mathcal{G}(K) = \mathcal{G}' \cup \left(\bigcup_{i=1}^k \mathcal{A}_i(K) \right).$$

Lemma 47

Given K sufficiently large, $\mathcal{g}$ has the following properties:

1. There are constants A and B with the following properties: suppose w is a $\mathcal{g}$ -geodesic. Let S_P be a horosphere with P conjugate to P_i . Suppose w begins and ends at X distance at most d from S_P . Then $w = xyz$, where $\ell(x) \leq Ad + B$, $\ell(z) \leq Ad + B$, and $y \in (\mathcal{A}_i(K))^*$.
2. If w begins on S_P , x is empty. If w ends on S_P , z is empty. In particular, a $\mathcal{g}$ -geodesic evaluating into P_i is written in letters all of which lie in P_i .
3. If we fix d then if w is sufficiently long, y is non-empty.

Proof—We claim that there is a bound r independent of K so that if e is a $g(K)$ edge which does not lie in P , then the X length of $r_P(e)$ is less than r . If e is an $\mathcal{A}_i(K)$ edge which does not lie in P , then it lies within a bounded distance of some horosphere other than S_P . By property 9 of Review 42, $r_{S_P}(S_{P'})$ has bounded diameter. The set of g' letters does not vary with K , and it is not hard to see that their edges also have bounded retractions onto S_P . This gives the bound r .

Now consider the case of a geodesic w which begins and ends in P . We wish to show that all edges of w lie in P . If this fails, we replace w with a sub-segment whose only contact with P are its two endpoints, p and q . Notice that it must therefore have length at least 2 since it leaves and returns to S_P . Let d_1 be the maximum distance from S_P to P . Then the path $r_P(w)$ starts and ends within distance d_1 of w . Thus $\ell(w) \geq \frac{d_X(p,q) - 2d_1}{r}$. Now consider an X -geodesic from p to q . This has length at most $d_X(p, q) + 2d_1$ and each point of it lies within distance d_1 of P . It follows that the $\mathcal{A}_i(K)$ distance between p and q is at most $\frac{d_X(p,q) + 2d_1}{K - d_1} + 1$. Choosing K sufficiently large contradicts the assumption that w was geodesic.

Now consider the case in which p and q do not necessarily lie on S_P . Let p'' and q'' be their respective projections onto S_P and p' and q' be points of P near these. There are λ and ε depending on K so that the embedding of G into X is a (λ, ε) -quasi-isometry. Consider tuv with t a geodesic from p to p' , u a geodesic from p' to q' and v a geodesic from q' to q . Then

$$\ell(w) \leq \ell(tuv) \leq 2\lambda d + 2\varepsilon + \frac{d_X(p', q') + 2d_1}{K - d_1} + 1.$$

Now if y does not appear in w , i.e., w contains no subword lying in P , then

$$\frac{d_X(p', q') - 2d_1}{r} + 1 \leq \ell(w) \leq 2\lambda d + 2\varepsilon + \frac{d_X(p', q') - 2d_1}{K + d_1} + 1.$$

The value of λ can only decrease as K increases, since λ measures how many $g(K)$ letters it takes to travel a certain distance in X , and for K sufficiently large, $K - d_1 > r$. Thus, for any sufficiently large K , there is a linear bound $\ell(w) < A'd + B'$ on those w for which y is empty.

We now suppose $w = xyz$ where y is the maximal portion of w lying in P and is non-empty. Let p''' and q''' be the endpoints of y . We claim that these must lie a bounded distance from p' and q' . To see this, notice that w is an X -quasi-geodesic. It follows from Lemma 43 that w fellow travels its X geodesic union the horosphere's that these meet. It is not hard to see that if $d_X(p, q)$ is sufficiently large, this X geodesic meets S_P near p'' and q'' .

For $i \neq j$, P_i and P_j meet in a finite (perhaps trivial) subgroup. We will assume that K is chosen large enough so that any non-trivial elements common to one or more subgroups appear as generators. After choosing K , we will refer to $g(K)$ and $\mathcal{A}_i(K)$ as g and $\mathcal{A}_i$.

We are now in a position to describe the Cannon's algorithm of Theorem 37. This depends on constants D and E . For each i , let $(\mathcal{A}_i, \mathbb{A}_i, S_i)$ be a D -tight Cannon's algorithm for P_i . We will assume that any rules operating inside a common subgroup rewrite immediately to a single letter and thus, these rules agree between the different S_i . We take S_g to be a collection of local geodesic rules which contain a left-hand side for each g word of length less than or equal to E which is not a geodesic. We assume that these agree with any rules which also appear in

some S_i . We will assume $D \geq E$. We take $\mathbb{A} = \mathcal{G} \cup (\cup_i \mathbb{A}_i)$ and $S = S_{\mathcal{G}} \cup (\cup_i S_i)$. We will show that with E sufficiently large, $\mathcal{D} = (\mathcal{G}, \mathbb{A}, S)$ is a Cannon's algorithm. This requires a series of lemmas.

We first check that the parabolic subgroup sub-Cannon's algorithms are still effectively D -tight within $\mathcal{D}$.

Lemma 48

Suppose that w is the result of $\mathcal{D}$ reducing a $\mathcal{G}$ input word and that u is a maximal $\mathbb{A}_i$ subword of w . Then u is a reduced word for a D -geodesic Cannon's algorithm for P_i .

Proof—Consider the process by which u is produced. Any non- $\mathbb{A}_i$ input letters which are consumed in the production of u must first be turned into $\mathbb{A}_i$ letters prior to their consumption by S_i . This can be done either by S_j writing $\mathbb{A}_j$ letters into the common subgroup $P_i \cap P_j$ with $i \neq j$, or, as the result of $S_{\mathcal{G}}$ rules rewriting a non-geodesic as a geodesic which lies in P_i . Other than rewriting as a single generator in the common subgroup, no S_j , $j \neq i$ can apply in the creation of u . Therefore, u could have been produced by applying the $S_{\mathcal{G}}$ rules and S_i rules to an input word in $\mathcal{A}_i^*$. The result now follows from the assumption that the Cannon's algorithm for P_i is tight.

It now follows that if w is the result of $\mathcal{D}$ -reducing a $\mathcal{G}^*$ input word, then w consists of reduced words from the parabolic subgroups alternating with E -local geodesics which do not contain any parabolic letters. These E -local geodesics may be empty, but by assuming that the parabolic subwords are maximal, we may assume that no two adjacent parabolic subwords lie in the same parabolic subgroup. Note that if two or more P_i meet in a non-trivial finite subgroup, any ambiguity where one parabolic subgroup ends and another begins can only consist of a single letter.

We will choose to decompose w in a slightly different manner. We choose a parameter $F < D$. We decompose w as

$$w = g_0 p_1 \dots g_{m-1} p_m g_m$$

where the p_j are the maximal parabolic subwords which represent group elements of length greater than F . Since all other maximal parabolic subwords represent group elements of length less than or equal to D , each is an $\mathbb{A}_i$ -geodesic for some i . It follows that the g_j are E -local geodesics. Again, some of the g_j may be empty, but not if they lie between P_i words.

Lemma 49

For D, E sufficiently large, there is $(\lambda, \varepsilon) = (\lambda_{D,E,F}, \varepsilon_{D,E,F})$ such that each g_i is a (λ, ε) -quasi-geodesic in H . While increasing F weakens the quasi-geodesicity, increasing D and E does not.

Proof—Let v be an E -local geodesic of length E . This is a Cayley graph geodesic, and hence an $X(\lambda, 0)$ -quasi geodesic, with λ depending only on the embedding of Γ into X . By Lemma 43, u is confined to a bounded neighborhood of H geodesic γ together with any horospheres that γ enters. Now γ cannot stray far into any horosphere, for otherwise u would contain parabolic subwords of length greater than F . This bounds the ratio between the X length and the H length of γ . Notice that this bound is independent of E . Thus, by increasing E , we proportionally increase the X -length of u . That is to say, there is (λ', ε') such that if u is a Cayley graph geodesic containing no parabolic subword of length greater than F , then u is an H -(λ', ε')-quasi-geodesic.

It is a standard result for δ -hyperbolic spaces that given (λ', ε') , for E sufficiently large, there is (λ, ε) so that every E -local (λ', ε') -quasigeodesic is a (λ, ε) -quasigeodesic. Thus, choosing E (and hence, D) sufficiently large makes each g_i an $H(\lambda, \varepsilon)$ -quasigeodesic as required.

Consider the decomposition of w into

$$w = g_0 p_1, \dots, g_{m-1} p_m g_m$$

as above. Ultimately, we must show that w is empty if and only if the input word which created it represents the identity. We will examine several paths related to w , namely

$$\begin{aligned}\sigma &= \sigma(w, F) = \gamma_0 \pi_1, \dots, \gamma_{m-1} \pi_m \gamma_m \\ \pi &= \pi(w, F) = g_0 \pi_1, \dots, g_{m-1} \pi_m g_m \\ \nu &= \nu(w, F) = g_0 q_1, \dots, g_{m-1} q_m g_m\end{aligned}$$

where

- Each γ_i is the H -geodesic for the corresponding g_i .
- Each π_i is the H -geodesic for the corresponding p_i .
- Each q_i is a Cayley graph geodesic for the corresponding p_i .

Lemma 50

Given D, E, F sufficiently large,

- There is (λ, ε) such that σ is an $H(\lambda, \varepsilon)$ -quasigeodesic. Increasing D and E does not worsen this quasigeodesicity.
- There is (λ, ε) such that π is an $H(\lambda, \varepsilon)$ -quasigeodesic.
- There is (λ, ε) such that ν is a Cayley graph (λ, ε) -quasigeodesic.

Proof—We first consider σ . We choose F sufficiently large. Since each π_i is long, by property 10 of Review 42, it spends only a limited time in a neighborhood of the exterior of its horoball. On the other hand, each γ_i can only spend a bounded time in the neighborhood of the horoballs it starts and ends at, for otherwise, by Lemma 47, it would start or end in the corresponding parabolic letters, contradicting the maximality of p_{i-1} (at its beginning) or p_i (at its end). Thus, the only way, σ can fail to satisfy the assumptions of Proposition 45 is if one or more of the γ_i is short, i.e., of X length less than $\delta' + 1$. In this case, we modify σ to produce σ' by deleting each short γ_i and replacing π_i with π'_i starting at the beginning of γ_i . Clearly σ and σ' asynchronously fellow travel. By Proposition 45, σ' is an H quasigeodesic, and thus, so is σ .

It now follows by Proposition 46 that π is an H quasigeodesic.

Finally, it follows from Lemma 44 that ν is an X quasigeodesic and hence a Cayley graph quasigeodesic.

In the case where each $P_i = \langle A_i \rangle$ has the falsification by fellow traveler property, this gives [22, Lemma 4.7]. It then follows that the language of geodesics in $G = \langle \mathcal{G} \rangle$ is a regular language and that the growth of $G = \langle \mathcal{G} \rangle$ is rational. This includes the limit groups of [25] since, as [9] has shown, these are hyperbolic relative to abelian subgroups.

Proof (Theorem 37)—We suppose that w is the result of $\mathcal{D}$ -reducing an input word in $v \in \mathcal{g}^*$. We must show that w is empty if and only if v represents the identity. Since w remembers its group element, the “only if” part is clear.

Suppose now that v represents the identity. Then $\sigma(w)$ is an H -quasigeodesic. Since it represents the identity, this bounds its length. This, in turn bounds the length of w . Recall that increasing D and E does not worsen the quasi-geodesicity of σ , and thus does not degrade the bound on the length of w . We may then assume that D and E are greater than this bound. Thus, w is a geodesic, in particular, a geodesic for the identity, and thus empty as required.

Proof (Corollary 40)—Let $G = \pi_1(M)$ where M is a finite volume negatively curved manifold with curvature bounded below and bounded away from 0. By [13], G is hyperbolic relative to nilpotent subgroups $P_1, \dots, P_k$. Now each P_i has a Cannon’s algorithm by Theorem 36. However, there is no guarantee that this is N -tight for P_i . It is, however, N -tight for matrix group $U_n(\mathbb{Z})$. Given any finite generating set $\mathcal{P}$ for P_i , we may include these into a generating set for $U_n(\mathbb{Z})$. Now, if $N \geq 1$, any N -tight Cannon’s algorithm for $U_n(\mathbb{Z})$ is 1-tight. It follows that for each $p \in \mathcal{P}$, p is the unique reduced word for itself. In particular, this is a 1-tight Cannon’s algorithm for P_i .

Since Lemma 47 holds for any sufficiently large K , we can assume that $\mathcal{A}_i$ contains any finite subset of P_i we select.

Now consider the paths of Lemma 50. The decompositions depend on a parameter, F , and this parameter is stated in terms of Cayley graph length. However, $\mathcal{A}_i$ is only used to ensure that each π_j is long, i.e., that the H geodesic of this group $\mathcal{A}_i$ element is long. By choice of K and hence, $\mathcal{A}_i$, we can force this to be the case for any parabolic group element whose reduced word is at least two letters long. The proof now proceeds as before.

6. Histories, Compression, Splitting and Splicing

This and the following section are devoted to showing that certain groups do not have Cannon’s algorithms. In this section we develop tools which appear broadly applicable to length-reducing rewriting systems. Thus we will be able to show that a particular group G has neither a Cannon’s algorithm, nor a non-incremental Cannon’s algorithm. Indeed, these methods and results have now been extended to the non-deterministic case in [17].

Let $w_0, \dots, w_n$ be the sequence of words produced as a rewriting algorithm makes n substitutions on w_0 . We call this sequence the *history to time n of w_0* . We can draw a *diagram* of the history as follows. Draw w_0 as a row of $\ell(w_0)$ adjacent unit squares, labelled with the letters of w_0 . For each $i > 0$ we draw w_i below w_{i-1} as follows. Draw a line segment under the first left-hand side appearing in w_{i-1} . (We call this a *substitution line*.) Underneath it put a row of equal width, height 1 rectangles, labelled with the corresponding right-hand side, or if the right-hand side is empty, put a single black rectangle. Fill the remainder of the row with a copy of whatever appears in that part of w_{i-1} .

The *width of a letter of w_i* is the width of its rectangle in the diagram. The *width of a subword of w_i* , not to be confused with its length, is the sum of the widths of the letters making it up (i.e. disregarding any black rectangles).

In order to get a handle on how the number of letters in a word decreases as the algorithm runs, we consider how the widths of letters increase.

Lemma 51

Let W be the length of the longest left-hand side of the rewriting system. In a diagram, the letters of any right-hand side (under a substitution line) have width at least $W/(W-1)$ times that of the narrowest letter in the corresponding left-hand side (above the substitution line).

Proof—If we were to first make all the letters of the left-hand side equal in width, deleting any black rectangles which appear, we would certainly not make the narrowest letter any narrower. Then at least one letter, of at most W , is removed, giving a further expansion of at least the stated factor.

Next we define the *generation* of each letter in a diagram. The generation of each letter in the first row is 0. The generation of a letter in row $i > 0$ is the generation of the letter above it, if it is not in a right-hand side, or one more than the least generation of the letters above the substitution line, if it is in a right-hand side.

Lemma 52

If the generation of a letter is n then its width is at least $(\frac{W}{W-1})^n$.

Proof—True for row 0. Suppose it is true for row $i-1$. Since each letter in row i not in a right-hand side has the same generation and width as the corresponding letter in the row above, the assertion holds for these letters. By Lemma 51, any letter in a right-hand side is at least $W/(W-1)$ times the width of the narrowest letter in the corresponding left-hand side. But the generation of each right-hand side letter exceeds the generation of the narrowest left-hand side letter by *at most* one. Since the assertion is assumed to hold for the narrowest letter in the left-hand side, it holds for the letters of the right-hand side.

Definition 53 (See Fig. 1.)

A *splitting path* of length n in a diagram for $w_0 \dots w_t$ consists of n vertical line segments running between letters, from the top of the diagram to the bottom, such that successive segments either join end to end, or are linked by a substitution line. Segments may not cut substitution lines. For each segment substitution lines between the top and bottom of the segment, all lie to the same side of the segment.

Lemma 54

If w_t contains a letter of generation g , then the diagram contains a splitting path of length at most $2g+2$ ending next to the letter. We may choose the path to end on either side of it.

Proof—Start at the bottom of the diagram with a vertical segment next to the letter of generation g . Extend upward until we come to a substitution line. Above that line will be a letter of generation $g-1$. Start a new segment next to that letter and continue on up. After hitting at most g substitution lines we reach the top of the diagram. (If we hit an endpoint of a substitution line we start a new segment only if the letter we are following is under the line.)

This is not yet a splitting path: our vertical segments could still have substitution lines on both sides. When this happens it can only be with substitutions to the left in the upper part of the segment and to the right in the lower. (A sequence of substitutions going right to left would have to cross the vertical segment because such substitutions always overlap.) We split each such segment at the appropriate point and we are done.

Associated with each splitting path are its *details*: for each vertical segment we record whether any substitutions take place to the left or the right. (If neither, we can arbitrarily designate it

as left.) For a left segment we record the first $W - 1$ letters to its right (which will be constant), or to the end of the word if nearer. For a right segment we record the $W - 1$ letters to the left, or to the start of the word if nearer. If a segment ends on a substitution line we record the left-hand side, the position at which the path splits it (in the range $0 - W$) and the position at which the next segment splits the right-hand side (in the range $0 - (W - 1)$).

We say that two splitting paths (in different diagrams) are *equivalent* if they have the same details. (Note: we do not require vertical segments to be the same height.)

For example, the details of the splitting path shown in Fig. 1 might be given as: (left, “aaa”), (right, “aaa”, “aaab”, 3, 2), (left, “aaa”), (right, “baa”, “aaab”, 2, 3), (left, “aa”).

Remark 55

There are no more than $(2(W + 1)^2/\mathbb{A}^{2W+1})^{n+1}$ equivalence classes of splitting path of length less than or equal to n .

Given a splitting path for $w_0, \dots, w_t$, we define w_i^- and w_i^+ to be the subwords of w_i , to the left and the right respectively of the path. The next lemma can be interpreted as telling us that the detail of the splitting path is like a message that is passed between w_0^- and w_0^+ : if v_0^- sends the same message as w_0^- , w_0^+ will not notice the change.

Lemma 56

Let $v_0, \dots, v_r$ and $w_0, \dots, w_s$ contain equivalent splitting paths. Then the history of $v_0^- w_0^+$, up to a suitable time, contains an equivalent splitting path, and ends with the word $v_r^- w_s^+$.

Proof—Cut the histories of v_0 and w_0 along their respective splitting paths. Fit the left half of v_0 's history with the right half of w_0 's history. The lengths of vertical segments are most likely unequal: one side or the other is constant so we just make as many copies of the constant side as required to fit the two together.

We claim that in the resulting sequence of words, each word differs from the next by replacing a left-hand side with its corresponding right-hand side. This is clear when both words lie on the same segment or on successive segments joined end-to-end. In the remaining case, both path details record the same left-hand side, split at the same point, and identical splitting points in the corresponding right-hand side.

We still have to show that the left-hand sides at which changes occur are those that would be chosen by the algorithm. Consider words joined at a left segment. The left-hand side begins in v_i^- (and ends in it as well, unless it is one of the left-hand sides on the path). Therefore, from the start of the left-hand side, the next W letters are the same whether v_i^- is completed by v_i^+ or w_j^+ (since these begin with the same $W - 1$ letters). The algorithm will therefore substitute at the same place in either word. Now consider words joined at a right segment. The left-hand side ends somewhere in w_j^+ . Any left-hand side in $v_i^- w_j^+$, starting to the left of this one, would have to start within $W - 1$ letters of the end of v_i^- , for otherwise it would be a left-hand side in v_i (wholly to the left of a right segment). But in view of this, the same left-hand side would appear in w_j (since w_j^- and v_i^- end with the same $W - 1$ letters).

It follows that we have constructed the history of $v_0^- w_0^+$. That it contains a copy of the same splitting path, and ends with $v_r^- w_s^+$, is clear.

We would like to be able to say that v_r^- is determined by v_0^- and the splitting path but unfortunately this is not quite true. Let $[v_r^-]$ denote the first word to the left of the last segment in the splitting path. What the proof of Lemma 56 shows is that $[v_r^-]$ is determined by v_0^- and the splitting path. If the last segment is a right segment, then $v_r^- = [v_r^-]$, but if not the best we can say is that v_r^- is obtained from $[v_r^-]$ by substitutions entirely inside the latter. Similar statements hold for w_s^+ .

6.1. Subwords and border letters

We extend the results of this section to subwords. If $v_0, \dots, v_t$ is a history of v_0 , and w_0 is a subword of v_0 , how shall we define the history of w_0 ? We can do it by fixing a *deletion convention* for the rewriting system: for each left-hand side, decide which letters are deleted and which are changed to get the corresponding right-hand side. It is then determined, when a substitution takes place over the boundary of $w_i \subset v_i$, which letters of the right-hand side belong to w_{i+1} and which do not. More generally we can consider v_0 to be split up into arbitrarily many subwords; a deletion convention will determine how each v_i is to be split up.

We want to define the diagram of v_0 's history in such a way that each subword gets its own "sub-diagram". In other words, we want the history of $w_0 \subseteq v_0$ to occupy a rectangular block underneath w_0 . Therefore, when a substitution takes place over a subword boundary, we adjust the widths of the right-hand side letters on either side of the new boundary to keep it vertically aligned under the previous boundary. The problem that arises is that a deletion may occur on one side only of the subword boundary: in that case Lemma 51 fails. We make the following adjustments.

We designate the $W - 1$ letters to either side of a subword boundary as *border letters* (see Fig. 2). When a right-hand side contains both border and non-border letters, we assign widths as follows. The number of border letters will be the same in the right-hand side as in the left-hand side, so we line them up under the border letters of the left-hand side and keep their widths the same; we expand the non-border letters to fill the remaining space evenly. Otherwise we assign widths as previously stated. Note that when a left-hand side contains a subword boundary, the right-hand side will consist only of border letters. Now Lemma 51 holds for all non-border letters.

We restrict the definition of the generation of a letter to non-border letters: the *generation* of a non-border letter in a right-hand side is one more than the least generation of the non-border letters of the corresponding left-hand side. With this adjustment, Lemma 52 goes through.

Splitting paths are defined as before except that we forbid any of the words included in the detail to cross a subword boundary. Lemma 54 gives us such a path since we follow the edges of non-border letters (letters for which the generation is defined). Clearly a splitting path for $w_0 \subseteq v_0$ is also one for v_0 .

We show that the number of splitting paths required to split all subword histories, with a given starting length, is bounded by a polynomial function of that starting length. This bound is independent of the total number of substitutions in the history. It follows that if we have enough histories, two of them will have equivalent splitting paths.

Lemma 57

Let w_0 be a subword of v_0 of length $\ell(w_0) = N$, and let $w_0, \dots, w_t$ be such that $\ell(w_t) \geq 2W - 1$. Then w_t has a splitting path in one of at most $C_1 N^{C_2}$ equivalence classes, where C_1, C_2 are positive constants depending only on $|\mathbb{A}|$ and W . The splitting path can be chosen to end next to any non-border letter of w_t .

Proof—Since $\ell(w_t) \geq 2W - 1$ it contains at least one non-border letter. Choose one, and let g be its generation. By Lemma 52, its width is at least $(\frac{W}{W-1})^g$. But since this cannot exceed $N = \ell(w_0)$, the width of w_0 , we have $g \leq \log_{(\frac{W}{W-1})} N$.

By Lemma 54, we can find a splitting path, ending next to our chosen letter, of length at most $2g + 2$. By Remark 55, the number of classes of splitting path, of length $\leq 2g + 2$, does not exceed $(2(W + 1)^2/\mathbb{A}^{2W+1})^{2g+3}$. Since g is bounded by a logarithm of N , the result follows.

We consider now a word divided into two subwords u_0, v_0 . We keep v_0 fixed, vary u_0 , and run the algorithm for some amount of time. Intuitively speaking, our algorithm carries information between the two subwords, giving in principle a number of possible values for v_t which is exponential in $\ell(v_0)$. We show that the number of distinct v_t that can actually arise is only polynomial in $\ell(v_0)$.

Lemma 58

Let v_0 be a fixed word of length $N \geq 1$ in $\mathbb{A}^*$. For each word u_0 we choose a time t and let $u_t v_t$ be the result of applying t substitutions to $u_0 v_0$; v_t is then a function of u_0 . There exist positive constants C_0, C , depending only on $|\mathbb{A}|$ and W , such that $v_t(u_0)$ takes at most $C_0 N^C$ distinct values as u_0 varies. The same bound applies if we instead define $v_t u_t$ to be the result of applying t substitutions to $v_0 u_0$.

Proof—First consider all v_t such that $\ell(v_t) \geq 2W - 1$. Then by Lemma 57, each $v_0, \dots, v_t$ has a splitting path ending $W - 1$ letters from the start of v_t , in one of at most $C_1 N^{C_2}$ classes.

Since $[v_t^+]$ is determined by v_0^+ and the class of the splitting path, and v_0^+ has at most N possible values (each being a subword at the end of v_0), $[v_t^+]$ takes at most $C_1 N^{C_2+1}$ distinct values as u_0 varies. Since v_t^+ is one of the (at most) $\ell([v_t^+]) \leq N$, words obtained by making substitutions in $[v_t^+]$, v_t^+ itself can take at most $C_1 N^{C_2+2}$ values. On the other hand, since $\ell(v_t^-) = W - 1$, this can take at most $|\mathbb{A}|^{W-1}$ values. Multiplying these two gives the required bound on the number of values v_t can take when $\ell(v_t) \geq 2W - 1$.

The number of possible words of length less than $2W - 1$ is constant and, since we are assuming $N \geq 1$, we can absorb this into C_0 .

The proof for $v_t u_t$ is similar.

Remark 59

If the rewriting system were not required to delete a letter with every substitution, the number of values $v_t(u_0)$ could take might well be exponential in $\ell(v_0)$.

7. Groups Which Have No Cannon's Algorithm

In this section we use the results of the previous section to exhibit groups which have neither a Cannon's algorithm nor non-incremental Cannon's algorithm. These results have been extended to the non-deterministic case in [17].

Theorem 60

Let G be a group with some fixed generating set. Suppose that for each $n \geq 0$ there are sets $S_1(n) \subset G$ and $S_2(n) \subset G$ satisfying the following:

1. For $i = 1, 2$ each element of $S_i(n)$ can be represented by a word of exactly length n .

2. There are $\alpha_0 > 0$ and $\alpha_1 > 1$ so that for infinitely many n

$$|S_i(n)| \geq \alpha_0 \alpha_1^n.$$

3. Each element of S_1 commutes with each element of S_2 .

Then G has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof—Suppose to the contrary that we have a deterministic Cannon's algorithm for G . Let $\mathbb{A}$ be the working alphabet and let W be the length of the longest left-hand side. Choose $n > 0$ such that $|S_i(n)| \geq \alpha_0 \alpha_1^n$, and

$$\frac{1}{2} \alpha_0 \alpha_1^n > C_1 n^{C_2+2} |\mathbb{A}|^{6W} C_0 n^C,$$

where C , C_0 , C_1 and C_2 are as in Lemmas 57 and 58. Let T_i be a set of words of length n representing $S_i(n)$, for $i = 1, 2$.

We shall consider the effect of our supposed Cannon's algorithm on words of the form $u_0 v_0 u_0^{-1} v_0^{-1}$, for $u_0 \in T_1$ and $v_0 \in T_2$. All such words must reduce to the empty word since $S_1(n)$ and $S_2(n)$ commute. Put $x_0 = u_0^{-1}$ and $y_0 = v_0^{-1}$, and let $u_t v_t x_t y_t$ denote the result of applying t substitutions to $u_0 v_0 x_0 y_0$.

Define t , as a function of u_0 and v_0 , to be the least integer $t \geq 0$ such that $\max\{\ell(v_t), \ell(x_t)\} < 3W$. That is, we run the algorithm until the first time at which both v_0 and x_0 have length less than $3W$. Then u_t , v_t , x_t and y_t are all well-defined functions of u_0 and v_0 . Since at most W letters are deleted in each step it follows that $2W \leq \max\{\ell(v_t), \ell(x_t)\} \leq 3W - 1$.

For each pair $(u_0, v_0) \in T_1 \times T_2$, one or both of the inequalities $\ell(v_t) \geq \ell(x_t)$, $\ell(x_t) \geq \ell(v_t)$ holds. Therefore one of these inequalities must hold for at least half of $T_1 \times T_2$. We shall suppose it is the first, and argue to obtain a contradiction; were it the second, a similar argument, interchanging the roles of v_t and x_t , would give a contradiction instead.

Step 1 (Fix a v_0): Since we are assuming that, for at least half the pairs (u_0, v_0) , $\ell(v_t) \geq \ell(x_t)$, we can certainly find a $v_0 \in T_2$ such that for at least half of $u_0 \in T_1$, $\ell(v_t(u_0, v_0)) \geq \ell(x_t(u_0, v_0))$. We fix this v_0 and henceforth regard u_t , v_t , x_t , y_t as functions of u_0 alone. Let $U = \{u_0 \in T_1 \mid \ell(v_t) \geq \ell(x_t)\}$. By the choices we have made, $|U| \geq \frac{1}{2}|T_1| \geq \frac{1}{2}\alpha_0 \alpha_1^n$.

Step 2 (Split the v_t using boundedly many splitting classes): By the definitions of t and U , $\ell(v_t) \geq 2W$ for each $u_0 \in U$. By Lemma 57, we can choose a splitting path for each $v_0, \dots, v_t$, using at most $C_1 n^{C_2}$ classes. If we take into account also the position in v_0 at which the path begins, and the position in v_t at which the path ends, we get at most $C_1 n^{C_2+2}$ classes.

Step 3 (The map $u_0 \mapsto v_t x_t y_t$ is many-to-one): By the definition of t , $\ell(v_t), \ell(x_t) \leq 3W - 1$, so the number of possible values $v_t x_t$ can take, as u_0 ranges over U , is less than $|\mathbb{A}|^{6W}$. By Lemma 58, the number of values y_t can take is at most $C_0 n^C$, for positive constants C_0 and C depending only on $|\mathbb{A}|$ and W . On the other hand $|U| \geq \frac{1}{2}\alpha_0 \alpha_1^n$ so, by our choice of n , $|U| > C_1 n^{C_2+2} |\mathbb{A}|^{6W} C_0 n^C$. It follows that there exists a set of at least $C_1 n^{C_2+2}$ u_0 's in U which all give the same $v_t x_t y_t$.

Step 4 (Construct a word that breaks the Cannon's algorithm): From Step 3, we have more than $C_1 n^{C_2+2}$ u_0 's giving the same $v_t x_t y_t$. From Step 2, we have at most $C_1 n^{C_2+2}$ positioned splitting path classes for v_t . Therefore we can find $u_0, u'_0 \in U$ such that (writing v'_t for $v_t(u'_0)$, etc.), $v_t x_t y_t = v'_t x'_t y'_t$ and $v_0, \dots, v_t$ and $v'_0, \dots, v'_t$ contain equivalent splitting paths, starting at the same position in $v_0 = v'_0$ and ending at the same position in $v_t = v'_t$.

By Lemma 56, running the algorithm on $u_0 v_0^{-1} v'_0 x'_0 y'_0$ yields $u_t v_t^{-1} v'_t x'_t y'_t$. Now $u_0 v_0^{-1} v'_0 x'_0 y'_0 = u_0 v_0 x'_0 y_0$, which does not represent the identity in G but rather $u_0 u_0'^{-1}$. On the other hand, $u_t v_t^{-1} v'_t x'_t y'_t = u_t v_t x_t y_t$ which reduces to the empty word. Therefore this rewriting system does not implement a Cannon's algorithm for G .

In fact, it is not hard to strengthen this to the following.

Theorem 61

Let G be a group with some fixed generating set. Suppose that for each $n \geq 0$ there are sets $S_1(n) \subset G$ and $S_2(n) \subset G$ satisfying the following:

1. For $i = 1, 2$ each element of $S_i(n)$ can be represented by a word of exactly length n .
2. There are $\alpha_0 > 0$ and $\alpha_1 > 1$ and $\alpha_2 > 0$ so that for all n

$$|S_1(n)| \geq \alpha_0 \alpha_1^n$$

and

$$|S_2(n)| \geq \alpha_2 n.$$

3. Each element of S_1 commutes with each element of S_2 .

Then G has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof—The proof is very similar to that of Theorem 60 except that we have to start with v_0 longer than u_0 . As before, suppose that we have a deterministic Cannon's algorithm for G , with working alphabet $\mathbb{A}$, and longest left-hand side of length W . Choose n_1 and n_2 such that

$$\frac{1}{2} \alpha_0 \alpha_1^{n_1} > C_1 n_2^{C_2+2} |\mathbb{A}|^{6W} C_0 n_2^C, \quad (1)$$

and

$$\frac{1}{2} \alpha_2 n_2 > C_1 n_1^{C_2+2} |\mathbb{A}|^{6W} C_0 n_1^C. \quad (2)$$

Let T_i be a set of words of length n_i bijecting to $S_i(n_i)$, for $i = 1, 2$.

As before, let $u_t v_t x_t y_t$ denote the result of applying t substitutions to $u_0 v_0 u_0^{-1} v_0^{-1}$. Define t , as a function of u_0, v_0 , such that $2W \leq \max\{\ell(v_t), \ell(x_t)\} \leq 3W - 1$.

If, for at least half of $(u_0, v_0) \in T_1 \times T_2$, $\ell(v_t) \geq \ell(x_t)$, we can argue as in Steps 1–4, using (1), to find u_0, u'_0 which break the algorithm. In the other case, arguing similarly, using (2), we can find v_0, v'_0 which break the algorithm.

Theorem 62

Suppose G has exponential growth and the center of G contains an infinite cyclic group. Then G has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof

We take among our generators for G a letter z denoting a central element of infinite order and a letter r denoting the identity. We can then take $S_1(n) = B(n)$, for if u is a geodesic denoting an element of length $k \leq n$, we can denote this element by ur^{n-k} . Thus each element of $B(n)$ is represented by a word of length n . Likewise, we can take $S_2 = \{z^k \mid |k| \leq n\}$. We then apply Theorem 61.

Corollary 63

$F_2 \times \mathbb{Z}$ has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Corollary 64

If G is a braid group of 3 or more strands, G has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Corollary 65

If M is a graph manifold one of whose pieces is a non-closed Seifert fibered space, then $\pi_1(M)$ has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Corollary 66

If M is a closed 3-manifold modeled on either $\mathbb{H}^2 \times \mathbb{R}$ or $\text{PSL}_2(\mathbb{R})$, then $\pi_1(M)$ has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Corollary 67

Thompson's group F has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof—Thompson's group F has exponential growth [8] and contains a subgroup isomorphic to the direct product of two copies of itself.

We will say that a subgroup A of G has exponential growth in G if $A \cap B(n)$ has exponential growth.

Theorem 68

Suppose G has an abelian subgroup which has exponential growth in G . Then G has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof—In this case we take $S_1(n) = S_2(n) = A \cap B(n)$ and apply Theorem 60.

Corollary 69

If G is a Baumslag–Solitar group

$$\langle a, t|ta^p t^{-1}=a^q \rangle$$

with $p \neq \pm q$ then G has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof—It is not hard to see that $\ell(a^n) = O(\ln n)$ and consequently, $\langle a \rangle$ has exponential growth in G .

Corollary 70

Suppose M is a closed 3-manifold modeled on solvegeometry. Then $\pi_1(M)$ has neither a Cannon's algorithm nor a non-incremental Cannon's algorithm.

Proof—In this case $\pi_1(M)$ contains a finite index subgroup of the form $A \rtimes \mathbb{Z}$ where A is isomorphic to $\mathbb{Z}^2$ and the action of the generator of $\mathbb{Z}$ has eigenvalues λ and λ^{-1} with the modulus of λ greater than 1. It follows that A has exponential growth in $\pi_1(M)$.

Combining these with our results on virtually nilpotent and geometrically finite groups we have:

Theorem 71

Suppose M is a graph manifold. Then $\pi_1(M)$ has a Cannon's algorithm if and only if none of the following hold:

1. M is closed $\mathbb{H}^2 \times \mathbb{R}$, $\mathrm{PSL}_2(\mathbb{R})$ or solvegeometry manifold, or
2. M has a non-closed Seifert fibered piece.

Acknowledgments

We wish to thank Gilbert Baumslag, Jason Behrstock, Brian Bowditch, Martin Bridson, Bill Floyd, Swarup Gadde, Bob Gilman, Susan Hermiller, Craig Hodgson, Chuck Miller, Walter Neumann and Kim Ruane for helpful conversations. We also wish to give special thanks to Jim Cannon for suggesting the key idea of this work to us during a conference at the ANU in 1996, and for working with us during the evolution of this paper.

References

1. Alonso, JM.; Brady, T.; Cooper, D.; Ferlini, V.; Lustig, M.; Mihalik, M.; Shapiro, M.; Short, H. Notes on word hyperbolic groups. In: Ghys, E.; Haefliger, A.; Verjovsky, A., editors. *Group Theory From a Geometric Viewpoint*. World Scientific; 1991.
2. Baumslag G, Miller CF, Short H. Unsolvability problems about small cancellation and word hyperbolic groups. *Bulletin of the London Math Soc* 1994;26:97–101.
3. Bowditch, B. Relatively hyperbolic groups. 1999. <http://www.maths.soton.ac.uk/staff/Bowditch/papers/bhb-relhyp.pdf>
4. Buntrock G, Otto F. Growing context-sensitive languages and Church–Rosser languages. *Inform Comput* 1998;141(1):1–36.
5. Cannon J. The combinatorial structure of cocompact discrete hyperbolic groups. *Geom Dedicata* 1984;16(2):123–148.
6. Cannon, J. The theory of negatively curved spaces and groups. In: Bedford, T.; Keane, M.; Series, C., editors. *Ergodic Theory, Symbolic Dynamics and Hyperbolic Spaces*. Oxford University Press; 1991. p. 315–369.
7. Cannon J. private communication.
8. Cannon JW, Floyd WJ, Parry WR. Introductory notes on Richard Thompson's groups. *Enseign Math* 1996;42(2):215–256.

9. Dahmani F. Combination of convergence groups. *Geom Topol* 2003;7:933–963.
10. Dehn, M. *Papers on Group Theory and Topology*. Springer; New York: 1987.
11. Domanski B, Anshel M. The complexity of Dehn’s algorithm for word problems in groups. *J Algorithms* 1985;6:543–549.
12. Drutu, C.; Sapir, M. Tree-graded spaces and asymptotic cones of groups. 2007. <http://www.citebase.org/abstract?id=oai:arXiv.org:math/0405030v3>
13. Farb B. Relatively hyperbolic groups. *GAFA* 1998;8:810–840.
14. Gromov M. Groups of polynomial growth and expanding maps (with an appendix by Jacques Tits). *Publications Mathématiques de l’IHÉS* 1981;53:53–78.
15. Holt DF. Word-hyperbolic groups have real-time word problem. *Int J Algebra Comput* 2000;10(2): 221–227.
16. Holt DF, Rees S. Solving the word problem in real time. *J London Math Soc* 2001;63:623–639.
17. Holt DF, Rees S, Shapiro M. Groups that do and do not have context sensitive word problem, to appear. *International Journal of Algebra and Computation*.
18. Hopcroft, J.; Ullman, J. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley; 1979.
19. Kambites M. personal communications.
20. Kambites, M.; Otto, F. Church-Rosser groups and growing context-sensitive groups. available from <http://www.maths.manchester.ac.uk/~kambites/>
21. Lysionok IG. On some algorithmic properties of hyperbolic groups. *Izv Akad Nauk Ser Math* 1989;53 (4)English translation. *Math USSR Izv* 1990;35:145–163.
22. Neumann WD, Shapiro M. Automatic structures, rational growth and geometrically finite hyperbolic groups. *Inventiones Mathematicae* 1995;120:259–287.
23. Osin, DV. *Relatively hyperbolic groups: Intrinsic geometry, algebraic properties, and algorithmic problems*. American Mathematical Society; 2006.
24. Segal, D. *Polycyclic Groups*. Cambridge University Press; 1983.
25. Sela Z. Diophantine geometry over groups I: Makanin-Razborov diagrams. *IHES Publ Math* 2001;93:31–105.
26. Shapiro M. A note on context sensitive languages and word problems. *Int J Algebra Comput* 1994;4 (4):493–497.

a_0	a_0	a_0	a_0	a_0	a_0	b_0	a_0	a_0	a_0	a_0	a_0	a_0	a_0	c_0
a_0	a_0	a_0	b_1	a_1	a_1	a_0	a_0	a_0	a_0	a_0	a_0	a_0	a_0	c_0
b_1	a_1	a_1	a_1	a_1	a_1	a_0	a_0	a_0	a_0	a_0	a_0	a_0	a_0	c_0
b_1	a_1	a_1	a_1	a_1	a_1	a_0	a_0	a_0	a_0	b_1	a_1	b_1		
b_1	a_1	a_1	a_1	a_1	a_1	a_0	b_1	a_1	a_1	a_1	a_1	b_1		
b_1	a_1	a_1	b_1	a_1	a_1	a_1	a_1	a_1	a_1	a_1	a_1	b_1		
b_2	b_2	a_2	a_1	a_1	a_1	a_1	a_1	a_1	a_1	a_1	a_1	b_1		
b_2	b_2	a_2	a_1	a_1	a_1	b_2	a_2	a_2						
b_2	b_2	b_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2		
c_3	b_3	a_3	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2	a_2		

Fig. 1.

A diagram for one possible history of a rewriting system. The numbers indicate generations. Dotted lines indicate a splitting path.

a	a	a	a	a	a ₀	a ₀	a ₀	a ₀	b ₀	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	a ₀	a ₀	a ₀	b ₀	a ₀	a ₀	a ₀	c ₀
a	a	a	a	a	a ₀	b ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	a ₀	a ₀	a ₀	b ₀	a ₀	a ₀	a ₀	c ₀
a	a	a	b	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	a ₀	a ₀	a ₀	b ₀	a ₀	a ₀	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	a ₀	a ₀	a ₀	b ₀	a ₀	a ₀	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	a ₀	a ₀	a ₀	b ₁	a ₁	a ₁	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	b	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	b	a	a	a	a	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₀	a	a	a	a	a	a	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	b ₁	a	a	a	a	a	a	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	a	a	a	a	a ₁	a ₁	a ₁	a ₁	a ₁	a ₀	a ₀	a ₀	a ₀	a ₁	a	a	a	a	a	a	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	a	a	b	a	a ₂	a ₂	a ₂	a ₂	a ₂	a ₁	a ₁	a ₁	a ₁	a ₁	a	a	a	a	a	a	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	b	a	a	a	a ₂	a ₂	a ₂	a ₂	a ₂	a ₁	a ₁	a ₁	a ₁	a ₁	a	a	a	a	a	a	a	a	a ₁	a ₁	a ₀	a ₀	c ₀
b	b	a	a	a	a ₂	a ₂	a ₂	a ₂	a ₂	a ₁	a ₁	a ₁	a ₁	a ₁	a	a	a	a	a	a	a	a	a ₁	a ₁	b ₁	a ₁	b ₁
b	b	a	a	a	a ₂	a ₂	a ₂	a ₂	a ₂	a ₁	a ₁	a ₁	a ₁	a ₁	a	a	a	a	a	a	a	b	a ₂	a ₂	a ₁	b ₁	b ₁
b	b	a	a	a	a ₂	a ₂	a ₂	a ₂	a ₂	a ₁	a ₁	a ₁	a ₁	a ₁	a	a	b	a	a	a	a	a	a ₂	a ₂	a ₁	b ₁	b ₁

Fig. 2.

A diagram for a history with subwords. Solid vertical lines indicate subword boundaries. Cross-hatching indicates border letters.