A Strong Generating Test and Short Presentations for Permutation Groups

Gene Cooperman*x and Larry Finkelstein†

College of Computer Science
Northeastern University
360 Huntington Ave.
Boston, Mass. 02115

* The work of this author was supported in part by the National Science Foundation under grant number CCR-8903952.
† The work of this author was supported in part by the National Science Foundation under grant numbers DCR-8603293 and CCR-8903952.

Abstract

The group membership problem for permutation groups is perhaps the most important problem of computational group theory. Solution of this problem seems to depend intrinsically on constructing a strong generating set. Until now, recognizing if a set of generators is strong has been thought to be as hard as constructing a strong generating set from an arbitrary generating set. This paper shows how to verify a strong generating set in $O(n^4)$ time, where n is the size of the set on which the group acts. This is faster than the best known algorithms in the literature. The work also leads to related algorithms for discovering all orbit information contained in an arbitrary set of generators S in $O(n|S| + n \log n)$ time, and, if S is strong, for finding a presentation with no more than $|S|(n-1)$ relations. Refinements in the analysis are given for the case in which a small base exists.

1. Introduction.

Let G be a permutation group acting on an n-element set Ω and let G be specified by a list S of generating permutations. A fundamental issue in many computational group theory algorithms, as well as in applications, is deciding if S is a strong generating set. If S has this property, then it is possible to efficiently perform computations such as testing membership of arbitrary permutations in G.

Sims (1971), gave the first efficient algorithm for constructing a strong generating. A good description of Sims' original idea together with a discussion of implementation issues is given by Butler & Cannon (1982). Other interesting versions which are variations of Sims' original method are given by Leon (1980), Knuth (1981), and Jerrum (1986). A novel approach due to Babai *et al.* (1988) has worst case asymptotic running time of $O(n^4 \log^c(n))$ which appears to be the best so far presented, but has not yet been fully implemented.

Sims' idea is to use the original generating set S to construct a data structure for computing a family U of coset representatives for the subgroups in the point stabilizer sequence of G (relative to a fixed ordering of Ω), which are implied by the generators of S. The algorithm then attempts to discover if U is *complete* or, equivalently, if S is a strong generating set by attempting to express certain elements $g \in G$, known as Schreier generators, as a product of elements of U of the form

$$g = u_{i_k} u_{i_{k-1}} \cdots u_{i_1}$$

where $u_{i_j} \in U$, u_{i_j} fixes the first $i_j - 1$ points of Ω, and u_{i_j} moves the ${i_j}^{\text{th}}$ point. In this case, we say that g *factors* through U. If each Schreier generator factors through U, then U is complete. Otherwise, additional generators are added to S, U is updated to reflect the change in S, and new Schreier generators are checked to see whether they factor through U. Usually, U will be complete after only a few additional generators are added to S. However, this can not be assured until one has checked that each Schreier generator can be factored. This checking phase dominates the running time.

Our first result, Theorem 4.3, is a test for whether S is a strong generating set. If $S^{(i)}$ is the subset of S which fixes the first $i-1$ points of Ω, and $m = |\{i \colon S^{(i)} - S^{(i+1)} \neq \emptyset\}|$, then our test takes time $O(mn|S| + mn^2 \min(m \log(n), n-1))$. In particular, since $m \leq n-1$, our test takes time $O(n^4)$. This represents an improvement on the worst case performance for existing tests, which usually require as much time as to construct a strong generating set.

In order to briefly describe the key idea, first let U be a family of coset representatives implied by S for the subgroups in the point stabilizer sequence for G. A traditional test involves checking if each of the $O(|S|n^2)$ Schreier generators factors through U. Since factoring requires $O(n^2)$ time, this would realize a test in $O(|S|n^4)$ time. The novelty of our approach, is the introduction of an alternative set of $O(|S|n)$ generators, called *basic generators* which can be used in the same way for testing if S is a strong generating set. The reduction in the size of the test set of generators by a factor of $O(n)$ leads to a corresponding reduction in the running time for the test.

Section 3 contains the heart of the proof. It shows that if each basic generator factors through U, then any element of G factors through U as well. This can be proven by purely combinatorial techniques on an abstract group, H, generated by Γ. A factorization of a basic generator yields an equation in G, with the basic generator on the left, and an equivalent factored word on the right. The basic generators can be expressed as words in Γ, and the equivalent factored words can be expressed in terms of coset representatives, each of which may also be expressed as words in Γ. Thus the factorization equations determine a congruence relation on words in Γ. Theorem 3.1 shows, under certain technical hypotheses, that all elements of this abstract group are congruent to factored words.

There is a fundamental connection between building a membership algorithm for permutation groups and deriving a presentation for these groups. The ability to express each element of G in a unique factored form leads fairly directly to a presentation for G. Conversely, some of the more interesting membership algorithms such as the Schreier-Todd-Coxeter-Sims algorithm (Leon, 1980) and the Babai-Luks-Seress algorithm (Babai *et al.*, 1988) explicitly use presentations to incrementally construct a data structure for testing group membership. The approach of this paper toward constructing presentations should be contrasted with the approach of Cannon (1973). Cannon usually finds shorter presentations than those of this paper, but no theoretical bounds are given on the number of relations, and the time for Cannon to compute the presentation is usually longer.

Our second result, Theorem 4.4 and Corollary 4.5, describes a method for constructing short presentations. If S is a strong generating set for G, then we are able to give a presentation for G using $|S|$ generators and $(n-1)|S|$ relations. Since we can always choose S such that $|S| \leq n-1$, this implies that every permutation group on n letters has a presentation with at most $n-1$ generators and $(n-1)^2$ relations. Furthermore, if G has a base of size m, then we can choose S such that $|S| \leq \min(n-1, m\log(n))$. This represents an improvement over the Babai-Luks-Seress algorithm which can be used to construct a presentation with $O(n^2 \log^c(n))$ relations.

Our third result, presented in section 2, is an algorithm for the construction of a *primitive data structure* which in turn can be used for the fast computation of two important data structures for storing a family U of coset representatives implied by S for the subgroups of the point stabilizer chain of G. These are the Schreier vector data structure (Sims, 1971) and the labelled branching data structure (Jerrum, 1986). Our algorithm constructs this new data structure in time $O(|S|n)$ using $O(|S|n)$ space. Furthermore, our algorithm returns a subset S' of S so that U can be derived from S' as well as S. From there, one can create a Schreier vector data structure in the same $O(|S|n)$, allowing cosets to be accessed in time $O(n^2)$ using $O(|S|n)$ storage, or else one can create a labelled branching in time $O(n^2)$, allowing cosets to be accessed in time $O(n)$, but using $O(n^2)$ storage. An interesting byproduct, is that if it is known beforehand that S is a strong generating set, then we can construct in time $O(|S|n)$, a subset S' of S which is a strong generating set as well, and which satisfies $|S'| \leq \min(n-1, \log|G|)$.

2. Group Membership Data Structures for Point Stabilizer Sequences.

Let G be a permutation group acting on an n-element set Ω and let $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ be an arbitrary ordering of the points of Ω. Let $G^{(1)} = G$ and let $G^{(i)}$, $1 < i \leq n$, be the subgroup of G consisting of all permutations of G which fixes each of the points $\alpha_1, \alpha_2, \ldots, \alpha_{i-1}$. Then the sequence

$$G = G^{(1)} \supseteq G^{(2)} \cdots \supseteq G^{(n)} = \{e\}$$

is called the *point stabilizer sequence* of G relative to α. A generating set S for G is a *strong generating set* if

$$\langle G^{(i)} \cap S \rangle = G^{(i)}, \quad 1 \leq i \leq n-1.$$

For each i, $1 \leq i \leq n-1$, let $U^{(i)}$ be a set of elements of $G^{(i)}$ which belong to different cosets of $G^{(i+1)}$. The set

$$U = \cup_{i=1}^{n-1} U^{(i)}$$

is called a *family of coset representatives* for the point stabilizer sequence of G relative to α. The associated cosets will be referred to as *cosets for the point stabilizer sequence.* Each set $U^{(i)}$ is in a 1-1 correspondence with a subset of points in the orbit of α_i under $G^{(i)}$, denoted ${\alpha_i}^{G^{(i)}}$, in the sense that each element of $U^{(i)}$ maps α_i to a distinct point of ${\alpha_i}^{G^{(i)}}$. U is said to be *complete* if $U^{(i)}$ is a complete set of coset representatives for $G^{(i+1)}$ in $G^{(i)}$ for each i, $1 \leq i \leq n-1$. We will always assume that each $U^{(i)}$ contains the identity element.

For each family of coset representatives U there is a uniquely defined function

$$\mu\colon \{(i,j)\colon 1 \leq i \leq j \leq n\} \to G \cup \{\texttt{NIL}\}$$

dependant on U, with the property that $\mu(i,i)$ is the identity element for all i and for $i < j$, $\mu(i,j) = \texttt{NIL}$ or $\mu(i,j)$ is an element of $U^{(i)}$ which moves α_i to α_j.

Given $g \in Sym(\Omega)$, a fundamental procedure in computational group theory is to attempt to factor g as a unique product of non-identity elements of U in the form,

$$g = \mu(i_k, j_k) \cdots \mu(i_2, j_2)\mu(i_1, j_1),$$

where $i_k > \cdots > i_2 > i_1$. If g can be written in this form, then we say that g *factors through* U.

Procedure Factor. *Input:* An element $g \in S_n$ and a family of coset representatives U for the point stabilizer sequence relative to an ordering α for G. *Output:* `TRUE` if g factors through U and `FALSE` otherwise.

```
Initialize $h$ to $g$
For $i \leftarrow 1$ to $n-1$ do
    Let $j$ be such that $\alpha_j = \alpha_i^h$
        If $\alpha_j \neq \alpha_i$ then
            If $\mu(i,j) \neq$ NIL then
                Set $h \leftarrow h\mu(i,j)^{-1}$
            Else return(FALSE)
Return(TRUE)
```

If U is complete, then Factor returns true if and only if $g \in G$. This provides an effective *membership test.*

It is evident from this discussion that it is crucial to have an efficient method for representing a family of coset representatives for G. Several interesting methods have emerged over the last two decades which represent a balance in the use of time and space. The least space efficient method is to store the family of coset representatives in an $n \times n$ matrix in which the (i,j) entry is set either to an element of $U^{(i)}$ which moves α_i to α_j or to `NIL` if no such element exists. In this case, $O(n^2)$ permutations are stored, but a given element of U can be recovered in constant time. In the case where G has a small base, the most space efficient method in practice is the Schreier vector data structure due to Sims (1971). This data structure stores words (or equivalently, pointers) in the generators. However, in this case, the algorithm used to construct a specific coset representative may require as many as $n-1$ multiplies in the worst case. An interesting alternative was described by Jerrum (1986) and will be referred to as a *labelled branching*.

One can easily conceptualize the Schreier vector data structure for G as a sequence of directed labelled trees, denoted $Schreier(i)$, $1 \leq i \leq n-1$, rooted at i. The nodes of $Schreier(i)$ are the indices j such that α_j is in the orbit ${\alpha_i}^{G^{(i)}}$ and whose edges are of the form (j,k) with label $s \in S \cap G^{(i)}$ where ${\alpha_j}^s = \alpha_k$. Given $j \in Schreier(i)$, one can construct a coset representative $p \in G^{(i)}$ which moves α_i to α_j by simply taking the product of the edge labels along the path from i to j. A good description of implementation issues for this data structure is given in Butler & Cannon (1982).

Formally, a *branching* on Ω is a directed forest with vertices $1,\dots,n$. A branching $\mathcal{B}$ is said to be a *labelled branching for G* relative to α, if each edge (i,j) is labelled by a permutation σ_{ij} so that the following property holds:

(i) $\sigma_{ij} \in G^{(i)}$ and moves α_i to α_j.

(ii) The set of edge labels of $\mathcal{B}$ generate G.

A labelled branching $\mathcal{B}$ is said to be *complete* if the following additional property holds:

(iii) If α_k is in the $G^{(i)}$ orbit of α_i, then there is a path in $\mathcal{B}$ from i to k.

The definitions are equivalent to those of Jerrum, although Jerrum does not use the word complete, and defines the branching and labels as independent data structures.

Criterion (iii) ensures that the edge labels of $\mathcal{B}$ form a strong generating set for G relative to the ordering α. This means that the set of edge labels of $\mathcal{B}$ which fix $\alpha_1, \alpha_2, \cdots, \alpha_{i-1}$ generates $G^{(i)}$. Criterion (iii) further ensures that we can find representatives for all cosets of the point stabilizer sequence of G with elements of the form, $\sigma_{i_0 i_1}\sigma_{i_1 i_2}\dots\sigma_{i_{k-1} i_k}$, for some sequence, $i_0 < i_1 < i_2 < \cdots < i_k$. Rather than actually storing the edge labels of $\mathcal{B}$, we store node labels $\tau[i]$ for each node α_i. We define $\tau[i]$ to be the product of the edge labels along the path from r to α_i where node r is the root of the subtree of $\mathcal{B}$ containing α_i. (If $r = \alpha_i$ then $\tau[i]$ is the identity.) It then follows that the product of edge labels from node α_i to node α_j may be realized as the single permutation multiply $\tau[i]^{-1}\tau[j]$. In particular, when $\mathcal{B}$ is complete, each coset entry for the point stabilizer sequence can be computed at the cost of one inversion and one multiplication.

These methods for representing a family of coset representatives for the point stabilizer sequence have the following features in common: they use an underlying data structure together with a simple algorithm for recovering a specific coset representative. In the algorithms to be presented, it will be convenient for us to express our ideas without reference to a specific data structure. Furthermore, in the course of attempting to construct a strong generating set, we may only know a subset of a complete family of coset representatives. With this in mind, we define a *group membership data structure* for G, to be an ordered pair $(\mathcal{G}, \mu)$ where $\mathcal{G}$ is a data structure which is used to define a family of coset representatives U for the point stabilizer sequence of G relative to a fixed ordering α (usually not mentioned). μ is a function defined in terms of $\mathcal{G}$ which is consistent with the definition of μ above.

As an example of a group membership data structure, consider the pair $(\mathcal{B}, \mu)$ where $\mathcal{B}$ is a labelled branching for G and $\mu(i,j) = \tau[i]^{-1}\tau[j]$ if there is a path in $\mathcal{B}$ from α_i to α_j and `NIL` otherwise. It is straightforward to interpret the other data structures in this form. We shall later introduce, in conjunction with our strong generating test, a group membership data structure which is a cross between the Schreier vector data structure and labelled branchings.

$(\mathcal{G}, \mu)$ is said to be *complete* if $\mathcal{G}$ defines a complete family of coset representatives for G. In this case, the set of points $\{\alpha_i\colon \exists j, j > i, \mu(i,j) \neq \texttt{NIL}\}$ forms a *base* for G. A base has the property that only the identity of G fixes each element of the base.

2.1. Fast Augmentation Algorithms.

Let S be a generating set for G. Define $S^{(i)} = S \cap G^{(i)}$ for $1 \leq i \leq n-1$. $(\mathcal{G}, \mu)$ is *fully augmented* with respect to S if the following condition holds:

For each $\alpha_j \in {\alpha_i}^{\langle S^{(i)}\rangle}$ such that $1 \leq i \leq n-1$, $\mu(i,j) \in \langle S^{(i)}\rangle$, and $\mu(i,j) = \texttt{NIL}$ otherwise.

In particular, if S is a strong generating set for G, then a group membership data structure fully augmented for S must define a complete family of coset representatives of G.

Two important issues which emerge in many permutation group algorithms are the construction of a group membership data structure $(\mathcal{G}, \mu)$ fully augmented for S and the elimination from S of redundant generators. We say that S has *redundant generators* if there is a strict subset S' of S so that $(\mathcal{G}, \mu)$ is fully augmented for S' as well. In this case, S and S' have the same *orbit information* for the point stabilizer sequence, i.e., ${\alpha_i}^{\langle S^{(i)}\rangle} = {\alpha_i}^{\langle S'^{(i)}\rangle}$, $1 \leq i \leq n-1$ If S is a strong generating set, then S' will also be a strong generating set. On the other hand if S is not known to be a strong generating, then there is no guarantee that S' will even generate G.

In this section, we describe a fast algorithm for solving both problems simultaneously. We will describe a procedure Augment which has input S and returns the reduced generating set S'. Augment also returns two data structures $\mathcal{I}$ and *parent* which are used to store the orbit information for ${\alpha_i}^{\langle S^{(i)}\rangle} = {\alpha_i}^{\langle S'^{(i)}\rangle}$, $1 \leq i \leq n-1$. Once $\mathcal{I}$ and *parent* have been constructed, it will be possible to efficiently create a labelled branching and Schreier vector data structure. This is accomplished in Procedures Build-Branch and Build-Schreier-Vector which are described in sections 2.1.2 and 2.1.3 respectively.

Our main result can be summarized in the following theorem.

Theorem 2.1. Let S be a generating set for G and let S' be the subset of S returned by Augment. Let $m = |\{i\colon S^{(i)} - S^{(i+1)} \neq \emptyset\}|$.

(i) *If S is a strong generating set, then S' is as well and $|S'| \leq \min(n-1, \log(|G|))$.*

(ii) *A Schreier vector data structure fully augmented for both S and S' can be constructed in time $O(|S|n + n\log(n))$ using $O(|S|n)$ space.*

(iii) *A labelled branching fully augmented for both S and S' can be constructed in time $O(|S|n+n^2)$ using $O(|S|n + n^2)$ space.*

The following result is an immediate corollary to Theorem 2.1(i) and the fact that $\log(|G|) = \log\prod_i |U^{(i)}| = \sum_i \log|U^{(i)}| \leq m\log(n)$, when U is complete. It will be of use in the strong generating test.

Corollary 2.2. *If $m = |\{i\colon S^{(i)} - S^{(i+1)} \neq \emptyset\}|$, then either* *(i) There exists a subset S' of S so that $|S'| \leq min(n-1, m\log(n))$ and both S' and S generate the same orbit information for the point stabilizer sequence, or*

(ii) S is not a strong generating set.

2.2. Procedure Augment.

For simplicity, assume that α is the identity permutation for the remainder of Section 2. The two important data structures created by Procedure Augment are $\mathcal{I}$ and *parent*. $\mathcal{I}$ is an acyclic (undirected) graph with each edge $\{i, j\}$ labelled by two elements: an element $\sigma_{ij} \in S \cup S^{-1}$ which moves i to j and $\sigma_{ji} = {\sigma_{ij}}^{-1}$. If $k = min(i, j)$, then we do not require that $\sigma_{ij} \in \langle S^{(k)}\rangle$. *parent* is an array of length n whose entries define the forest structure for a labelled branching fully augmented for S. The set S' will emerge as the subset of S used to label the edges of $\mathcal{I}$.

We will use two intermediate data structures, *component* and *orbit*, which are arrays of size n. The value of *component* is an index into the *orbit* array. *orbit* is an array whose elements are structures with the fields *link*, *root*, and *members*. Augment proceeds in a bottom up fashion decrementing i from $n-1$ downto 1. After the i^{th} iteration, the connected components of $\mathcal{I}$ represent the orbits of $\langle S^{(i)}\rangle$, *orbit*[*component*[j]].*members* is a linked list of the nodes in the orbit containing j and *orbit*[*component*[j]].*link* is a temporary link pointer used to merge orbits. For two nodes $j, k \in \Omega$, *orbit*[*component*[j]] = *orbit*[*component*[k]] if and only if *component*[j] = *component*[k]. Since the orbit members are represented as linked lists, two orbits' members can be merged in constant time. Since a node is a member of at most one orbit at any time, the total space for storing all orbit members should not exceed $O(n)$ (assuming constant space to store one node). *orbit*[*component*[j]].*root* is the smallest node in the $\langle S^{(i)}\rangle$ orbit containing j, and the product of the edge labels along the path from *orbit*[*component*[j]].*root* to j represents an element of $\langle S^{(i)}\rangle$ which moves *orbit*[*component*[j]].*root* to j. In particular, since *orbit*[*component*[i]].*root* $= i$, we will be able to set *parent*[j] for all nodes, $j > i$ such that *parent*[j] $\geq i$.

Procedure Augment. *Input:* A generating set S for G. *Output:* $\mathcal{I}$, *parent* and S' as described. *Intermediate data structures: component* and *orbit*. *Auxiliary Functions: Merge-Orbits.*

```
[Initialize $\mathcal{I}$, $orbit$ , $component$ and the $parent$ array.]
Initialize $parent$. $(\forall i, parent[i] \leftarrow$ NIL$)$
Initialize $\mathcal{I}$ to a trivial labelled graph on $\Omega$
Initialize $S' \leftarrow \emptyset$
Set $component[n] \leftarrow n$
Set $orbit[n].members \leftarrow \{n\}$, $orbit[n].root \leftarrow n$
For $i \leftarrow n-1$ downto 1 do
    Set $component[i] \leftarrow i$
    Set $orbit[i].members \leftarrow \{i\}$, $orbit[i].root \leftarrow i$
    If $S^{(i)} - S^{(i+1)} \neq \emptyset$ then
        For $j \geq i$, set $orbit[j].link \leftarrow$ NIL
    For $\rho \in S^{(i)} - S^{(i+1)}$ do
        [Merge components of $\mathcal{I}$ under $\rho$.]
        For $j \leftarrow i$ to $n-1$ do
            Set $k \leftarrow j^\rho$
            If $component[k] \neq component[j]$ then
                [$\rho$ is a new element of the reduced generating set.]
                Set $S' \leftarrow S' \cup \{\rho\}$
                [Update $parent$.]
            If $i \in \{orbit[component[j]].root, orbit[component[k]].root\}$ then
               Let $p \in \{orbit[component[j]].root, orbit[component[k]].root\} - \{i\}$
               For $p' \in orbit[component[p]].members$ such that $parent[p'] = \mathtt{NIL}$ do
                  Set $parent[k'] \leftarrow i$ in $\mathcal{B}$
            [$\sigma_{jk}$ and $\sigma_{kj}$ should just point to $\rho$ and $\rho^{-1}$, requiring constant time.]
            Add edge $\{j, k\}$ to $\mathcal{I}$, set $\sigma_{jk} \leftarrow \rho$ and $\sigma_{kj} \leftarrow \rho^{-1}$
            Set $orbit[component[j]].link \leftarrow orbit[component[j]].link \cup component[k]$
      If $S^{(i)} - S^{(i+1)} \neq \emptyset$, then Merge-Orbits($i$)
   Return($S', \mathcal{I}, parent$)
```

Procedure Merge-Orbits. *Input:* Level, i. *Side Effects:* $\mathcal{I}$ is modified so that all components with link pointer connecting them are destructively merged into a single component.

```
For $j \leftarrow i$ to $n-1$ do
   Set $closed[j] \leftarrow \mathtt{NIL}$
For $j \leftarrow i$ to $n-1$ do
   If $closed[component[j]] = \mathtt{NIL}$ then
      [Search through all components linked to $j$
       in time proportional to number of links]
      Set $openset \leftarrow \{component[j]\}$
      For $k \in openset$ do
         Set $openset \leftarrow openset - \{k\}$
         Set $closed[k] \leftarrow j$
         For $\ell \in orbit[k].link$ do
            If $closed[\ell] = \mathtt{NIL}$ then
               Set $openset \leftarrow openset \cup \{\ell\}$
For $j \leftarrow i$ to $n-1$ do
   If $closed[j] \neq \mathtt{NIL}$ then
      [Merge component $j$ with component $closed[j]$]
[The union is done by connecting linked lists in constant time.]
Set $orbit[closed[j].members] \leftarrow orbit[closed[j].members] \cup orbit[j].members$
Set $orbit[j].root \leftarrow orbit[closed[j].root]$
[Update $component$.]
For $k \in orbit[j].members$ do
   Set $component[k] \leftarrow j$
Set $orbit[j].members \leftarrow \mathtt{NIL}$
```

We first prove certain facts about $\mathcal{I}$, *parent* and *orbit* which are necessary for the proof of Theorem 2.1. We inductively define j to be a *descendant* of i if $j = i$ or if $parent[j]$ is a descendant of i. The following result is easily seen by induction, and the proof is omitted.

Lemma 2.3. *After the i^{th} iteration, but before the $i - 1^{\text{st}}$ iteration, the following properties hold for $\mathcal{I}$, $\mathcal{B}$ and orbit.*

(i) $\mathcal{I}$ is acyclic.

(ii) For each $j \geq i$, $orbit[component[j]].root$ is the smallest point in the $\langle S^{(i)} \rangle$ orbit containing j, there is a unique path in $\mathcal{I}$ from $orbit[component[j]].root$ to j, and the product of the edge labels along this path is an element of $\langle {S'}^{(i)} \rangle$ which moves $orbit[component[j]].root$ to j.

(iii) $parent[j] = i$ if and only if j is in $orbit[component[i]].members$ for each k such that $i < k < j$, but k is not in $orbit[component[i]].members$.

(iv) $orbit[component[j]].root = i$ if and only if j is a descendant of i and $parent[i] = \texttt{NIL}$.

The next result is an immediate corollary to Lemma 2.3(ii)-(iv).

Lemma 2.4. *The parent array determines a forest structure which is the same as that for a labelled branching fully augmented for both S and S'.*

Lemma 2.5. *Augment requires $O(|S|n)$ time and $O(|S|n)$ space.*

Proof: Only the time bound is discussed, since the space bound is clear. Each generator $\rho \in S$ is applied to each point of Ω only during that iteration of i for which $\rho \in S^{(i)} - S^{(i+1)}$. Thus there are at most $|S|$ non-trivial iterations. These $|S|$ iterations take time $O(|S|n)$.

Each time a new generator ρ is added to S', the components of $\mathcal{I}$ are merged in order to be compatible with the action of ρ. It takes $O(n)$ work to decide which components of $\mathcal{I}$ need to be merged, and each merger requires constant time to update the fields of *orbit*. Further, since each orbit can be merged into $orbit[component[i]]$ at most once by a given ρ, checking $orbit[component[j]].members$, $j \neq i$ for nodes with *parent* set to NIL can cost at most $O(n)$ time. Since S' is a subset of S, the total time for Augment, aside from the time to update *component* is $O(|S|n)$.

Finally, the cost of updating *component* over all calls to *Merge-Orbits* is at most $O(|S|n)$. This is clear since *Merge-Orbits* is called at most $|S|$ times, and does $O(n)$ work.

Proof of Theorem 2.1(i) We are given that S is a strong generating set for G and must show that S' is a strong generating set for G and $|S'| \leq \min(n-1, \log(|G|))$. The first assertion is an immediate consequence of Lemma 2.4.

To prove that $|S'| \leq \min(n-1, \log(|G|))$, first observe that S' is enlarged in Augment, only when two orbits are merged. Hence, $|S'| \leq n-1$. It suffices, therefore to show that $|S'| \leq \log(|G|)$.

Now

$$|G| = \prod_i |i^{G^{(i)}}|, \quad i \text{ a base point},$$

implies that

$$\log(|G|) = \sum_i \log(|i^{G^{(i)}}|), \quad i \text{ a base point}.$$

Therefore our assertion can be reduced to showing that for each base point i, Augment will add at most $\log(|i^{G^{(i)}}|)$ elements of S to S'.

Consider what happens when the main for loop is considering the base point i. At the beginning of the loop, we know that $\langle S' \rangle = G^{(i+1)}$ and the orbit of $\langle S' \rangle$ which contains i consists of $\{i\}$. Since S is a strong generating set, each new element $\rho \in S^{(i)} - S^{(i+1)}$ added to S' must enlarge $\langle S' \rangle$ but not enlarge $\langle S' \cap G^{(i+1)} \rangle = \langle S \cap G^{(i+1)} \rangle = G^{(i+1)}$. Hence $|i^{\langle S' \cup \{\rho\} \rangle}| > |i^{\langle S' \rangle}|$. Since $\langle S' \rangle \subseteq G^{(i+1)}$, $|i^{\langle S' \cup \{\rho\} \rangle}| = [\langle S' \cup \{\rho\} \rangle : \langle S' \rangle] |i^{\langle S' \rangle}|$. This implies $[\langle S' \cup \{\rho\} \rangle : \langle S' \rangle] > 1$. However, since the index is an integer, it is at least two. Therefore, each time we add a generator to S', we at least double the size of $i^{\langle S' \rangle}$. Since we stop when $i^{\langle S' \rangle} = i^{G^{(i)}}$, it follows that at most $\log(|i^{G^{(i)}}|)$ elements of $S^{(i)} - S^{(i+1)}$ can be added to S' as required. □

Remarks. directions.

(i) If the procedure is being used as part of a strong generating test, then it can be used to immediately recognize that certain generating sets are not strong. If $i^{\langle S' \rangle}$ does not grow by an integer factor, then S can not be a strong generating set. Further, if at the i^{th} level a newly added generator $\rho \in S' \cap (S^{(i+1)} - S^{(i)})$ does not merge *orbit*[*component*[i]], then S is not a strong generating set. These tests are not sufficient, since the two permutations, (1 2) and (1 2 3 4), are not a strong generating set, but would not be rejected by the above tests.

(ii) One may wish to use Procedure Augment to discover a small reduced generating set for S with the same orbit information as for S, even in the case that S is not strong. In that case, the loop for $\rho \in S^{(i)} - S^{(i+1)}$ should be replaced by two successive loops. The first loop should add to S' only those ρ that enlarge *orbit*[*component*[i]], although those ρ should continue to be used to merge all possible orbits. The second loop should add to S' the remaining ρ that merge orbits.

2.3. Construction of a Fully Augmented Schreier Vector.

For each point i such that $S^{(i)} - S^{(i+1)} \neq \emptyset$, *Schreier*($i$) can be constructed by a simple breadth first search from node i using the descendants of i. We will use the array *backptr* and *svector* as in (Butler & Cannon (1982)) to describe the tree. Of course, this code would have to be executed for each point i as above.

Procedure Build-Schreier-Vector *Input:* $\mathcal{I}$, *parent*, and i such that *parent*[i] $=$ `NIL` and $S^{(i)} - S^{(i+1)} \neq \emptyset$. *Output: Schreier*($i$) *as described by svector and backptr.*

```
For $i \leftarrow 1$ to $n$ do
    Set $svector[i] \leftarrow identity$ and $backptr[i] \leftarrow -1$
[Compute descendants of $i$.]
Set $descendant[i] \leftarrow$ TRUE
For $j \leftarrow i+1$ to $n$ do
    Set $descendant[j] \leftarrow$ NIL
For $j \leftarrow i+1$ to $n$ do
    If $parent[j] \neq$ NIL and $descendant[parent[j]] =$ TRUE then set $descendant[j] \leftarrow$ TRUE
[Use breadth first search in $\mathcal{I}$ to compute backptr and svector.]
Set $open\_set \leftarrow \{i\}$
For $j \in open\_set$ do
    Remove $j$ from $open\_set$ and set $descendant[j] \leftarrow$ NIL
    Let $\mathcal{J}$ be the set of nodes adjacent to $j$ in $\mathcal{I}$
    For each $k \in \mathcal{J}$ such that $descendant[k] =$ TRUE do
        Set $svector[k] \leftarrow \sigma_{jk}$, $backptr[k] \leftarrow j$ and add $k$ to $open\_set$
Return(svector, backptr)
```

Lemma 2.6. *Build-Schreier-Vector correctly computes svector and backptr for Schreier*(i).

Proof: It is clear by induction that there is a path in $\mathcal{I}$ from i to k through j. Since i is a root node, Lemma 2.3(ii) implies that the path is unique, and *svector*[k] $\in G^{(i)}$. Finally, *svector*[k] moves j to k by definition of σ_{jk}. The correctness of *backptr* follows trivially by induction. ◻

Proof of Theorem 2.1(ii).

We will show that the construction of the m Schreier vectors takes time $O(mn)$. Recalling that $m = |\{i: S^{(i)} - S^{(i+1)} \neq \emptyset\}|$, it is clear that $m \leq |S|$, and so the $O(mn)$ time to construct the Schreier vectors is within the bound of $O(|S|n)$ required to build $\mathcal{I}$ and *parent*. This will prove the result.

It is clear that initializing *backptr*, *svector* and *descendant* takes time $O(n)$. The key to showing that the breadth first search takes $O(n)$ time is the fact that $\mathcal{I}$ is acyclic. The sum of the sizes of the sets $\mathcal{J}$ computed for each node j visited is at most twice the number of edges of $\mathcal{I}$. So, $Schreier(i)$ can be built in linear time for each of the m points, i. □

2.4. Construction of a Fully Augmented Labelled Branching.

We will present an algorithm which shows how to use $\mathcal{I}$ and *parent* to obtain a labelled branching $\mathcal{B}$ which is fully augmented for S and S'. By Lemma 2.4, *parent* can be used to fill in the *parent* fields for the nodes of $\mathcal{B}$. Thus, it suffices to construct an array of permutations τ which can be used to complete the construction of $\mathcal{B}$.

Procedure Build-Branching *Input:* $\mathcal{I}$ and *parent*. *Output:* A labelled branching, $\mathcal{B}$, with node labels, τ. *Local Variables:* descendants is an array of length n.

```
[Set the forest structure of $\mathcal{B}$ according to the *parent* array.]
Let $\mathcal{B}$ be the trivial forest
Use *parent* to construct the parent fields for each node of $\mathcal{B}$.
[Initialize $\tau$.]
For $j \leftarrow 1$ to $n$ do
    If $parent[j] = \texttt{NIL}$ then set $\tau[j] \leftarrow identity$
For $i \leftarrow n$ downto 1 such that $(S^{(i)} - S^{(i+1)}) \neq \emptyset$ and $parent[i] = \texttt{NIL}$ do
    [Compute descendants of $i$.]
    Set $descendant[i] \leftarrow \texttt{TRUE}$
    For $j \leftarrow i+1$ to $n$ do
        Set $descendant[j] \leftarrow \texttt{NIL}$
    For $j \leftarrow i+1$ to $n$ do
        If $parent[j] \neq \texttt{NIL}$ and $descendant[parent[j]] = \texttt{TRUE}$ then set $descendant[j] \leftarrow \texttt{TRUE}$
    [Use breadth first search in $\mathcal{I}$ to compute $\tau[j]$.]
    Set $open\_set \leftarrow \{i\}$
    For $j \in open\_set$ do
        Remove $j$ from $open\_set$ and set $descendant[j] \leftarrow \texttt{NIL}$
        Let $\mathcal{J}$ be the set of nodes adjacent to $j$ in $\mathcal{I}$
        For each $k \in \mathcal{J}$ such that $descendant[k] = \texttt{TRUE}$ do
            Set $\tau[k] \leftarrow \tau[j]\sigma_{jk}$ and add $k$ to $open\_set$
```

Lemma 2.7. *Build-Branching correctly computes the τ fields of $\mathcal{B}$.*

Proof: Let r be an arbitrary root of $\mathcal{B}$, i.e. $parent[r] = \texttt{NIL}$. Then by Lemma 2.3(iv), at the end of the r^{th} iteration, the connected component $\mathcal{R}^{(r)}$ of $\mathcal{I}$ containing r consists of all nodes of $\mathcal{B}$ which are descendants of r. If $\mathcal{R}$ is the connected component containing r after $\mathcal{I}$ has been completely built, then $\mathcal{R}^{(r)}$ is a connected subgraph of $\mathcal{R}$. In particular, the breadth first search used to construct $\tau[i]$ for each descendant i of r searches precisely through $\mathcal{R}^{(r)}$. Thus if i is a descendant of r in $\mathcal{B}$, then i is a node of $\mathcal{R}^{(r)}$ and the procedure will fill in $\tau[i]$ with an element which moves r to i.

It remains to show that if i and j are descendants of r, with $i = parent[j]$, then $\tau[i]^{-1}\tau[j] \in G^{(i)}$. By Lemma 2.3(ii), there are unique simple paths in $\mathcal{I}$ from r to i and j. Since $\mathcal{I}$, and more specifically $\mathcal{R}$, is acyclic, there is a unique descendant k with the property that k is the intersection of the three simple paths, from r to i, from r to j, and from i to j. If we denote by ρ_{uv} the product of the edge labels along the path from u to v in $\mathcal{I}$, then it follows from the construction of τ, that

$\tau[i] = \tau[k]\rho_{ki}$ and $\tau[j] = \tau[k]\rho_{kj}$. Thus

$$\tau[i]^{-1}\tau[j] = (\tau[k]\rho_{ki})^{-1}(\tau[k]\rho_{kj}) = \rho_{ki}^{-1}\rho_{kj} = \rho_{ij}.$$

But $\rho_{ij} \in G^{(i)}$ by Lemma 2.3(ii),(iv). Thus the entries for the τ fields of $\mathcal{B}$ have the correct properties. □

Proof of Theorem 2.1(ii). It is clear that the construction of the τ fields for $\mathcal{B}$ in Build-Branching takes time $O(n^2)$. Since the initial construction of $\mathcal{I}$ and *parent* takes time $O(|S|n)$ by Lemma 2.5, it then follows that the final construction of a labelled branching fully augmented for S can be constructed in time $O(|S|n + n^2)$ using the procedure Build-Branching. □

3. Factorization in Monoids.

In this section, we extend the notion of factorization in permutation groups to a class of general monoids to be called *path monoids*. We will give sufficient conditions for a monoid in this class to be *factorizable*. Informally, this means that every element can be expressed by a word in "factored form". To do this will require introduction of the concepts of *path monoids* and *path products* on a path monoid. Our results on monoids will have important implications for both a strong generating test and presentations of finite permutation groups.

A *path monoid* is a finitely generated monoid, H, with associated 5-tuple $(\Gamma, n, h, \mathcal{U}, f)$. Γ is a finite set of generators for H, with associated *generator index map*,

$$h: \Gamma \to Z^+ \times Z^+,$$

and *maximum index*, n, such that if $\gamma \in \Gamma$ and $h(\gamma) = (i, j)$, then $1 \le i < j \le n$. We allow the possibility that $h(\gamma) = h(\gamma')$ for $\gamma \ne \gamma'$.

Given a path monoid, H, we define a chain of submonoids,

$$H = H^{(1)} \supseteq H^{(2)} \supseteq \cdots \supseteq H^{(n)} = \{\epsilon\},$$

where ϵ is the identity element. This is done by setting $H^{(i)} = \langle \Gamma^{(i)} \rangle$, where

$$\Gamma^{(i)} = \{\gamma \mid h(\gamma) = (r, s), \text{ and } i \le r < s \le n\}.$$

Considering the above chain of monoids as a formal analogue of the point stabilizer sequence for a permutation group, leads us to the generalization of a family of coset representatives for the point stabilizer sequence. The subset $\mathcal{U} \subseteq H$ is said to be the set of *path products* for the path monoid, H, with associated *path product index map*,

$$f: \mathcal{U} \to Z^+ \times Z^+,$$

such that the following holds:

(i) f is one-one, and for all $\rho \in \mathcal{U}$, if $f(\rho) = (i, j)$ then $1 \le i < j \le n$. (The $\rho \in \mathcal{U}$ will be denoted ρ_{ij}.)

(ii) If $\rho_{ij} \in \mathcal{U}$, then $\rho_{ij} \in H^{(i)} - H^{(i+1)}$.

(iii) If $\rho_{ij}, \rho_{jk} \in \mathcal{U}$, then there exists a path product $\rho_{ik} \in \mathcal{U}$.

(iv) If $\rho_{ik} \in \mathcal{U}$, $\rho_{jk} \in \mathcal{U}$ and $i < j$, then there exists a path product $\rho_{ij} \in \mathcal{U}$.

Let $\mathcal{U}$ be the set of path products. An *atomic path product* is a path product ρ_{ik}, such that there is no path product ρ_{jk} with $i < j < k$. It is easy to show using property (iv) that for any path product, ρ_{ij}, there are atomic path products $\rho_{i_0 i_1}, \rho_{i_1 i_2}, \ldots, \rho_{i_{m-1} i_m} \in \mathcal{U}$ with $i = i_0$ and $j = i_m$. We define $\mathcal{U}^{(k)} = \mathcal{U} \cap (H^{(k)} - H^{(k+1)})$. Hence, $\mathcal{U}^{(k)}$ consists of all elements of $\mathcal{U}$ of the form ρ_{km}, $k < m$. A *factored element* is the identity or any element of the form

$$\rho_{i_k j_k} \rho_{i_{k-1} j_{k-1}} \ldots \rho_{i_1 j_1},\ 1 \leq i_1 < i_2 \ldots < i_k < n.$$

An element in a path monoid H is *factorizable* under a congruence relation $\equiv$ if it is congruent to a factored element. By a *congruence* relation, we mean an equivalence relation on H satisfying the substitution axiom,

$$\text{(S)} \quad \text{If } v, w, w', x \in H \text{ and } w \equiv w', \text{ then } vwx \equiv vw'x.$$

The path monoid is *factorizable* under $\equiv$ if each element of H is factorizable under H.

Next, the slightly stronger hypothesis of proper factorizability is introduced. For $w \in H$, let i be the largest integer such that $w \in H^{(i)}$. w is said to *properly factor* if w is congruent to a factored element of $H^{(i)}$. Proper factorization is stronger than congruence to a general factored element of H. Since $H^{(n)} = \{e\}$, the identity element is considered to be properly factored. H is *properly factorizable* if each element of H can be properly factored.

Example 1.

An important example of a path monoid is a finite permutation group, G, with an associated 5-tuple (S, n, h_G, U^*, f_G). S is a finite set of generators not containing the identity. The maximum index is n, the number of points on which G acts. Let $h_G\colon S \to Z^+ \times Z^+$ be the generator index map for G defined by $h_G(g) = (i, j)$ where $g \in G^{(i)} - G^{(i+1)}$ and $j = i^g$.

Let U be a family of coset representatives for the point stabilizer sequence (not necessarily complete) and let $U^* = U - \{e\}$. Further, suppose $i^{U^{(i)}} = i^{\langle S \cap G^{(i)} \rangle}$. Then U^* is a set of path products for G, with path product index map $f_G\colon U^* \to Z^+ \times Z^+$, defined by $f_G(p) = (i, j)$, for $p \in U^*$, $p \in G^{(i)}$ and $i < j = i^p$.

Note that group equality is a congruence relation. If S is a strong generating set, then U is complete and so every element of G can be expressed as a factored element in U. In that case, G is factorizable under $=$ and is in fact properly factorizable. Also, whenever G is factorizable under $=$, every element can be expressed as a word in unique factored form.

Remark. Let $(\mathcal{G}, \mu)$ be a group membership data structure for G and let U be the family of coset representatives for the point stabilizer sequence of G defined by $\mathcal{G}$. Then $(\mathcal{G}, \mu)$ is fully augmented for S if and only if the condition $i^{U^{(i)}} = i^{\langle S \cap G^{(i)} \rangle}$ is satisfied for each i, $1 \leq i \leq n-1$.

Example 2.

We next show how a path monoid with m generators induces a path monoid structure on the free monoid on m generators. The induced structure is determined by the choice of a map between the two generating sets. Let G be a path monoid with 5-tuple (S, n, h_G, U, f_G). Let Γ be a finite set, with one-one and onto map, $\eta\colon \Gamma \to S$, and let H be the free monoid on Γ. η induces a path monoid structure on H with associated 5-tuple $(\Gamma, n, h_H, \mathcal{U}, f_H)$. h_H is a generator index map, $h_H\colon \Gamma \to Z^+ \times Z^+$, such that $h_H = h_G \circ \eta$. The maximum index, n, is the same for G and H. If we extend η to an epimorphism from H onto G, then η maps $H^{(i)}$ onto $G^{(i)}$ for $1 \leq i \leq n$.

To define $\mathcal{U}$, let p_{ij} be an arbitrary element of $U \cap (G^{(i)} - G^{(i+1)})$ and let $p_{ij} = s_{i_1} s_{i_2} \dots s_{i_\ell}$, where each $s_{i_k} \in S \cap G^{(i)}$ (otherwise condition (ii) of the definition of path products would be violated). The representation of p_{ij} in this form need not be unique, but such a representation does exist and once chosen, will remain fixed. Define $\mathcal{U} = \eta'(U)$ where $\eta'(p_{ij}) = \rho_{ij}$ is defined by replacing each s_{i_k} in the above representation for p_{ij} by the unique element $\gamma_{ik} = \eta^{-1}(s_{i_k}) \in \Gamma$. Note that $\eta \circ \eta'$ is the identity map on U. Given this definition, we can then define $f_H = f_G \circ \eta$. In particular, $f_H(\rho_{ij}) = f_G(p_{ij})$. It is straightforward to show that conditions (ii)-(iv) of the definition of path product are satisfied for $\mathcal{U}$ and f_H. For example, (ii) follows from the fact that η maps each $H^{(i)}$ onto $G^{(i)}$ and that $\eta(\rho_{ij}) = p_{ij} \in G^{(i)} - G^{(i+1)}$.

Next, assume that G is factorizable under $=$ and that each element of G can be written in unique form as a factored element. This, for example is the case where G is a permutation group as in Example 1. Let $\equiv_H$ be the congruence relation defined by $x \equiv_H y$ for $x, y \in H$ if $\eta(x) = \eta(y)$. Extend η' from a map of U into H to a map of G into H by writing each $g \in G$ in factored form $g = p_{i_k j_k} p_{i_{k-1} j_{k-1}} \cdots p_{i_1 j_1}$ where $i_k > i_{k-1} > \dots i_1$ and each $p_{i_\ell j_\ell} \in U$ and then setting $\eta'(g) = \eta'(p_{i_k j_k}) \eta'(p_{i_{k-1} j_{k-1}}) \dots \eta'(p_{i_1 j_1})$. Clearly η' is well defined and $\eta'(g)$ is a factored element in H. In particular, for each $x \in H$, $x \equiv_H \eta'(\eta(x))$ and $\eta'(\eta(x))$ is a factored element of H. Thus H is factorizable under $\equiv_H$ if G is factorizable under $=$. Similarly H is properly factorizable under $\equiv_H$ if G is properly factorizable under $=$. $\Box$

Our main result is that, subject to certain additional conditions which will usually be satisfied in our applications, H is properly factorizable if and only if a certain set of elements, called *basic generators* is properly factorizable. The set of basic generators, denoted $\mathcal{T}$, consists of all products of the form:

($\mathcal{T}_1$) $\rho_{ij}\gamma$, $\gamma \in \Gamma^{(j+1)}$, and ρ_{ij} is an atomic path product, or

($\mathcal{T}_2$) $\rho_{ij}\gamma$, $\gamma \in \Gamma$, $h(\gamma) = (k, l)$, an atomic path product ρ_{ii_1} exists with $i \le k < i_1$, and either $i_1 = j$ or there exists a path product $\rho_{i_1 j}$.

Remark. In fact the basic generators are precisely those words, w, on Γ such that w is not a factored word, and every proper subword of w is a factored word.

Theorem 3.1. *Let H be a path monoid with 5-tuple $(\Gamma, n, h, \mathcal{U}, f)$ and let $\equiv$ be a congruence relation in H. Assume that the following conditions hold:*

(i) For every $\gamma \in \Gamma$ such that $h(\gamma) = (i, j)$, there is a path product $\rho_{ij} \in \mathcal{U}$ such that $\gamma \equiv y\rho_{ij}$ for some $y \in H^{(i+1)}$.

(ii) If $\rho_{ij}, \rho_{jk} \in \mathcal{U}$, then $\rho_{ij}\rho_{jk} \equiv y\rho_{ik}$ for some $y \in H^{(i+1)}$.

(iii) If $\rho_{ik}, \rho_{jk} \in \mathcal{U}$ and $i < j$, then $\rho_{ik} \equiv y\rho_{ij}\rho_{jk}$ for some $y \in H^{(i+1)}$.

(iv) Each basic generator is properly factorizable, and

(v) If $\rho_{ij}\gamma$ is a basic generator of the form $\mathcal{T}_1$, then $\rho_{ij}\gamma \equiv y\rho_{ij}$ where $y \in H^{(i+1)}$ is a factored element.

Then, H is properly factorizable under $\equiv$.

We require three preliminary results, all of which are proved within the context of the hypotheses of Theorem 3.1. Let $\lambda_i(g)$ be the length of the shortest word in $\Gamma^{(i)}$ which equals g. If g is the identity, then $\lambda_i(g) = 0$ for all i. We shall refer to $\lambda_i(g)$ as the *length of g with respect to* $\Gamma^{(i)}$, or the *length of g* where it is clear with respect to which monoid. We will also say g is *shorter than* h if $\lambda_i(g) < \lambda_i(h)$. Many of the results will be proven by induction on this length.

Lemma 3.2. *Let ρ_{ij} be a path product and let $w \in H^{(j+1)}$. Then $\rho_{ij}w \equiv y\rho_{ij}$, for some $y \in H^{(i+1)}$.*

Proof. Assume the lemma does not hold, and ρ_{ij} and w form a counter-example for which the difference $j-i$ is minimal, and $\lambda_{j+1}(w)$ is minimal subject to fixing i and j. We first consider the case in which ρ_{ij} is an atomic path product. The result clearly holds for $\lambda_{j+1}(w) = 0$ (w the identity). If $w \in \Gamma^{(j+1)}$, then $\rho_{ij}w$ is a basic generator, and by hypothesis (v), $\rho_{ij}w \equiv y'\rho_{ij}$. Otherwise, let $w = \gamma w'$ for $\gamma \in \Gamma^{(j+1)}$, $w' \in H^{(j+1)}$, and $\lambda_{j+1}(w') = \lambda_{j+1}(w) - 1$. Then $\rho_{ij}w = \rho_{ij}\gamma w' \equiv y'\rho_{ij}w' \equiv y'y''\rho_{ij}$, where $y', y'' \in H^{(j+1)}$. The last equivalence follows because the length of w' is one less than that of w, and so $\rho_{ij}w'$ cannot be a counter-example to the lemma. The result now follows by setting $y = y'y''$.

Next, consider the case in which ρ_{ij} is non-atomic, and $\rho_{ij} = \rho_{ii'}\rho_{i'j}$ for $i < i' < j$. Hypothesis (iii) yields $\rho_{ij}w \equiv y'\rho_{ii'}\rho_{i'j}w$ for some $y' \in H^{(i+1)}$. Since $j - i' < j - i$, $\rho_{i'j}w$ is not a counter-example, and $y'\rho_{ii'}\rho_{i'j}w \equiv y'\rho_{ii'}z\rho_{i'j}$ for $z \in H^{(i'+1)}$. Since $i' - i < j - i$, $\rho_{ii'}z$ is not a counter-example and therefore, $\rho_{ii'}z \equiv z'\rho_{ii'}$, for $z' \in H^{(i+1)}$. Finally, using hypothesis (ii), we have $\rho_{ij}w \equiv y'z'\rho_{ii'}\rho_{i'j} \equiv y'z'z''\rho_{ij}$ for some $v' \in H^{(i+1)}$. The result now follows by setting $y = y'z'z''$. ∎

Lemma 3.3. *Given a path product ρ_{ij} and generator γ for H with $h(\gamma) = (j,k)$, there exist $y \in H^{(i+1)}$ and path product, ρ_{ik}, such that $\rho_{ij}\gamma \equiv y\rho_{ik}$.*

Proof. There is a $w \in H^{(j+1)} \subseteq H^{(i+1)}$ and $w', w'' \in H^{(i+1)}$ such that $\rho_{ij}\gamma \equiv \rho_{ij}w\rho_{jk} \equiv w'\rho_{ij}\rho_{jk} \equiv w'w''\rho_{ik}$. The three steps follow respectively from hypothesis (i), Lemma 3.2, and hypothesis (ii). The result follows by setting $y = w'w''$. ∎

The next result is the key to the proof of Theorem 3.1.

Lemma 3.4. *Let $\rho_{ij} \in \mathcal{U}$ and $w \in H^{(i)}$. Then $\rho_{ij}w \equiv y\rho$, where $y \in H^{(i+1)}$ and either ρ is the identity or $\rho = \rho_{ip}$ for some path product $\rho_{ip} \in \mathcal{U}$.*

Proof. The proof is by induction on the length of w. If $\lambda_i(w) = 0$ (w is the identity) then the result is clearly true. Assume therefore that $\lambda_i(w) > 0$ and that the result is true for $\tilde{w} \in H^{(i)}$ such that $\lambda_i(\tilde{w}) < \lambda_i(w)$. Set $w = \gamma w'$, where $\gamma \in \Gamma^{(i)}$, $w' \in H^{(i)}$ and $\lambda_i(w') = \lambda_i(w) - 1$.

Case I. Suppose first that $h(\gamma) = (j,s)$ for some s. By Lemma 3.3, $\rho_{ij}w = \rho_{ij}\gamma w' \equiv z\rho_{is}w'$ where $z \in H^{(i+1)}$. Since w' is shorter than w, the induction hypothesis is satisfied and we may write $\rho_{is}w' \equiv z'\rho$ where either ρ is the identity or $\rho = \rho_{ip}$ for some path product ρ_{ip}, and $z' \in H^{(i+1)}$. Thus $\rho_{ij}w \equiv zz'\rho$ where $zz' \in H^{(i+1)}$ and the result follows with $y = zz'$.

Case II. Next, suppose that $h(\gamma) = (r,s)$ with $j < r$. Then by Lemma 3.2, $\rho_{ij}\gamma \equiv z\rho_{ij}$ for some $z \in H^{(i+1)}$. Hence $\rho_{ij}w \equiv z\rho_{ij}w'$. Since w' is shorter than w, the induction hypothesis is satisfied and the proof follows as in the previous case.

Case III. We may therefore assume that $h(\gamma) = (r,s)$ with $j > r$. This is the hard case. Since $w \in H^{(i)}$ by assumption, $r \geq i$. As observed earlier, if ρ_{ij} is an arbitrary path product, then there exists a sequence of atomic path products $\rho_{i_0i_1}, \rho_{i_1i_2}, \ldots, \rho_{i_{m-1}i_m} \in \mathcal{U}$ with $i = i_0$ and $j = i_m$. Since $i \leq r < j$, there exists a unique index i_p such that $i_p \leq r < i_{p+1}$. Set $q = i_p$. Then the path product ρ_{qj} exists (by a simple consequence of the axioms for path products) and so $\rho_{qj}\gamma$ is a basic generator of type $\mathcal{T}_2$. If $i < q$, then by hypothesis (iii), $\rho_{ij} \equiv v'\rho_{iq}\rho_{qj}$ for some $v' \in H^{(i+1)}$. Otherwise, $i = q$. We can combine both cases by writing

$$\rho_{ij}\gamma w' \equiv v'\rho'\rho_{qj}\gamma w',$$

where either $\rho' = \rho_{iq} \in \mathcal{U}$, or ρ' is the identity in the case that $i = q$. Since basic generators properly factor by hypothesis (iv), $\rho_{qj}\gamma \equiv y'\rho''$ where either $y'\rho''$ is the identity or, for some $q' \geq q$, $\rho'' = \rho_{q'j'}$ and y' is a factored element in $H^{(q'+1)}$. Hence

$$\rho_{ij}\gamma w' \equiv v'\rho' y'\rho'' w'.$$

If ρ' or ρ'' is the identity, the proof can be completed as follows. If ρ' and ρ'' are both the identity, then $\rho_{ij}\gamma \equiv v'y'w'$, $v'y' \in H^{(i+1)}$. If $w' \in H^{(i+1)}$, then the result follows. Otherwise, w' has the form $w' = u\gamma' u'$, where $u \in H^{(i+1)}$, $\gamma' \in \Gamma^{(i)} - \Gamma^{(i+1)}$, and $u' \in H^{(i)}$ satisfies $\lambda_i(u') < \lambda_i(w') < \lambda_i(w)$. By hypothesis (i), we may write $w' \equiv u''\rho_{ij_1}u'$ with $u'' \in H^{(i+1)}$. The induction hypothesis applies to $\rho_{ij_1}u'$ and the proof may be completed as before. If ρ' is the identity and $\rho'' = \rho_{q'j'}$, $q < q'$, then $\rho_{ij}\gamma w' \equiv v'y'\rho'_{q'j'}w' \equiv yw'$, where $y' \in H^{(i+1)}$ and the proof follows as in the preceding case. Finally, if $\rho' = \rho_{iq}$ and ρ'' is the identity, then $\rho_{ij}\gamma w' \equiv v'\rho_{iq}y'w'$. The proof then follows as in previous cases.

Thus we may assume that both ρ' and ρ'' are not the identity, and so

$$\rho_{ij}\gamma w' \equiv v'\rho_{iq}y'\rho_{q'j'}w',$$

where $q \leq q'$ and $y' \in H^{(q'+1)}$. It follows from Lemma 3.2, that $\rho_{iq}y' \equiv y''\rho_{iq}$ where $y'' \in H^{(i+1)}$ and therefore,

$$\rho_{ij}\gamma w' \equiv v'y''\rho_{iq}\rho_{q'j'}w'.$$

If $q = q'$, then by hypothesis (ii), we may concatenate the path products to obtain

$$\rho_{ij}\gamma w' \equiv v'y''v''\rho_{ij'}w'$$

for $v'' \in H^{(i+1)}$. Otherwise, $q < q'$ and by Lemma 3.2, $\rho_{iq}\rho_{q'j'} \equiv y'''\rho_{iq}$ which yields

$$\rho_{ij}\gamma w' \equiv v'y''y'''\rho_{iq}w'$$

for $y''' \in H^{(i+1)}$. In either of the above instances, we may write

$$\rho_{ij}\gamma w' \equiv z\rho_{ip}w'$$

where $z \in H^{(i+1)}$. Since, w' is shorter than $w = \gamma_{rs}w'$, the induction hypothesis is satisfied and we may complete the proof as before. □

Proof of Theorem 3.1. We will prove the result by induction on $H^{(i+1)}$ as i goes from $n-1$ down to 0. In the base case, $H^{(n)}$ contains only the identity and the result clearly holds.

Let $0 \leq i < n$ and assume that all elements of $H^{(i+1)}$ can be properly factored. We will show that an arbitrary element w of $H^{(i)} - H^{(i+1)}$ can be properly factored.

Let $w = w'\gamma w''$ for $w' \in H^{(i+1)}$, $\gamma \in \Gamma^{(i)} - \Gamma^{(i+1)}$, and $w'' \in H^{(i)}$. By hypothesis (i), $\gamma \equiv y\rho_{ik}$ for some $y \in H^{(i+1)}$. Hence, $w \equiv w'y\rho_{ik}w''$. By Lemma 3.4, $\rho_{ik}w'' \equiv y'\rho_{mp}$ for $y' \in H^{(i+1)}$ and $m \geq i$ (or ρ_{mp} the identity). So, $w \equiv w'yy'\rho_{mp}$. If $m > i$ or ρ_{mp} is the identity, then $w'yy'\rho_{mp} \in H^{(i+1)}$ can be properly factored by induction.

If $m = i$, then $w'yy' \in H^{(i+1)}$ can be properly factored by the induction hypothesis, and it is congruent to a factored element, $z \in H^{(i+1)}$. Since $m = i$, $z\rho_{mp}$ is a factored element, and w is properly factorizable. So H is properly factorizable. □

Remark. Note that our use of the congruence relation was always to replace a basic generator by a factored word. Hence, one could define a rewriting system on a free monoid, whose rewrite rules each consist of a basic generator on the left hand side, and a factored word congruent to it on the right hand side. Under this interpretation, Theorem 3.1 gives sufficient conditions for existence of a complete rewriting system in which every word of a free monoid reduces to a factored word.

We conclude this section with a bound on the number of basic generators which will be useful in section 4.

Proposition 3.5. *The number of basic generators is at most* $(n-1)|\Gamma|$.

Proof. Each basic generator has the form $\rho_{ij}\gamma$, where ρ_{ij} is a path product and γ is a generator for a path monoid. We will show that each ordered pair (j,γ) uniquely determines at most one basic generator, $\rho_{ij}\gamma$. This immediately yields the bound on the number of relations, since $2 \le j \le n$.

We first observe that for each j, there exists at most one atomic path product of the form ρ_{ij}. In fact, if $i \ne i'$ and $\rho_{i'j}$ is also an atomic path product, then, assume without loss of generality that $i < i'$. It follows from condition (iv) of the definition of path products, that there exists a path product $\rho_{ii'}$. This contradicts ρ_{ij} being atomic.

There are two cases to consider, according to whether or not $\gamma \in \Gamma^{(j+1)}$. If $\gamma \in \Gamma^{(j+1)}$, then any basic generator, $\rho_{ij}\gamma$, must be of type $\mathcal{T}_1$, and ρ_{ij} must be atomic. Since for each j there is at most one atomic path product, there is at most one basic generator of the form $\rho_{ij}\gamma$.

In the second case, $\rho_{ij}\gamma$ must be of type $\mathcal{T}_2$. Let $h(\gamma) = (k,l)$, and note that $j > k$. Assume that there are two distinct indices, i and i', determining distinct basic generators: $\rho_{ij}\gamma$ and $\rho_{i'j}\gamma$. Let ρ_{ii_1} be an atomic path product such that $i \le k < i_1$ and either the path product ρ_{i_1j} exists or $i_1 = j$. Let $\rho_{i'i'_1}$ be a second atomic path product satisfying the analogous conditions.

We claim that $i_1 \ne i'_1$. To see, observe that if $i_1 = i'_1$ and $i < i'$, then condition (iv) of the definition of path products implies that the path product $\rho_{ii'}$ exists. This contradicts the assumption that ρ_{ii_1} is atomic. A similar contradiction is obtained if we assume that $i_1 = i'_1$ and $i' < i$.

Thus, without loss of generality, we may assume that $i_1 < i'_1$. We first establish that the path product $\rho_{i_1i'_1}$ exists. This is clear if $i'_1 = j$. Otherwise, $i_1 < i'_1 < j$ and ρ_{i_1j} and $\rho_{i'_1j}$ are path products implies, together with condition (iv) of the definition of path products, that such a path product exists. Now, $\rho_{i'i'_1}$ is atomic implies that $i_1 \le i'$. But then, $i \le k < i_1 \le i'$ contradicts the fact that $i' \le k$ as well. This concludes that there is at most one basic generator $\rho_{ij}\gamma$ of type $\mathcal{T}_2$ associated with (j,γ). ∎

4. Applications to Permutation Groups.

The main goal of this section is the application of Theorem 3.1 to obtain both a presentation for a permutation group and a criterion for deciding when a set of generators forms a strong generating set.

Let S be a generating set for the permutation group G, let $(\mathcal{G},\mu)$ be a group membership data structure for G, fully augmented for S, and let U be the family of coset representatives for the point stabilizer sequence of G defined by $\mathcal{G}$. We can consider G as a path monoid with associated 5-tuple (S, n, h_G, U^*, f_G) as in Example 1 of section 3. We do not require that U be complete. Let

$S^{(i)} = S \cap G^{(i)}$, $1 \le i \le n-1$. Let $T = T_1 \cup T_2$ be the set of basic generators for U and S.

(T_1) $p_{ij}g$, $g \in S^{(j+1)}$, and p_{ij} is an atomic path product, or
(T_2) $p_{ij}g$, $h(g) = (k,l)$, an atomic path product p_{ii_1} exists with $i \le k < i_1$, and either $i_1 = j$ or there exists a path product $p_{i_1 j}$,

where each $g \in S$ and each $p_{ij} \in U$.

The conditions of Theorem 3.1 may then be expressed as follows.

(R_1) If $g \in S^{(i)} - S^{(i+1)}$, then there exists a coset representative $p_{ij} \in U^*$ such that $g = wp_{ij}$ where $w \in \langle S^{(i+1)} \rangle$.

(R_2) If $p_{ij}, p_{jk} \in U^*$, then $p_{ij}p_{jk} = wp_{ik}$ where $w \in \langle S^{(i+1)} \rangle$.

(R_3) If $p_{ik}, p_{jk} \in U^*$ and $i < j$, then $p_{ik} = wp_{ij}p_{jk}$ where $w \in \langle S^{(i+1)} \rangle$.

(R_4) Every element of T factors with respect to U.

Theorem 4.1. *If hypotheses R_1–R_4 hold in G, then S is a strong generating set.*

Proof. Condition 5 of Theorem 3.1 follows immediately from condition R_4 and the fact that G is a permutation group. Thus, Theorem 3.1 applies, and G is properly factorizable. Hence, all elements of G properly factor. This means that if $g \in G^{(i)}$, then $g \in \cup_{k=i}^{n-1} U^{(k)} \subseteq \langle S^{(i)} \rangle$. Thus S is a strong generating set. □

It is important, from the point of view of an efficient implementation of the strong generating test, that the following property be satisfied for U.

(P) $p_{ij}p_{jk} = p_{ik}, \ \forall p_{ij}, p_{jk} \in U^*$

If (P) holds, then conditions (R_2) and (R_3) will always be true. The main advantage in using a labelled branching for $\mathcal{G}$ is to take advantage of the fact that (P) will then hold for U^*. In section 5 we will describe a space-efficient group membership data structure in which (P) holds for U^* and which is comparable in space requirements to the Schreier vector data structure.

Strong Generating Test. *Input: A generating set S for G. Output: A reduced strong generating set S' if S is a strong generating set and* `FALSE` *otherwise. Let $m = |\{i: S^{(i)} - S^{(i+1)} \neq \emptyset\}|$.*

(1) Let S' be a subset of S such that $|S'| \le min(n-1, m\log(n))$ and both S' and S generate the same orbit information for the point stabilizer sequence. If such a subset S' does not exist return `FALSE`. (Procedure Augment is an efficient way to either find such an S' or conclude that no such subset exists.)

(2) Build a labelled branching $\mathcal{B}$ which is fully augmented for S' and let U^* be the set of path products of $\mathcal{B}$. This gives rise to the path monoid described by the 5-tuple (S', n, h_G, U^*, f_G). (Procedure Build-Branching is an efficient method for constructing $\mathcal{B}$ when used in conjunction with Augment.)

(3) If any element of S does not factor through U, then return `FALSE`. (This guarantees that S' generates G.)

(4) If every basic generator factors through U, then return `TRUE`. Otherwise return `FALSE`. Here the basic generators are computed using the 5-tuple (S', n, h_G, U^*, f_G) for the path monoid G.

Theorem 4.2. *Let G be a permutation group on an n-element set and let S be set of generating permutations for G. Let $m = |\{i: S^{(i)} - S^{(i+1)} \neq \emptyset\}|$. Then it is possible to test in time $O(mn|S| + mn^2 \min(m\log(n), n-1))$ if S is a strong generating set for G.*

Proof: We first show that the strong generating test is correct. Step 1 is justified by Corollary 2.2. Thus either S' can be found which satisfies the stated conditions or S is not a strong generating set. In the case where S' is found, S is a strong generating set for G if and only if S' is as well. In Step 3, if $g \in S$ does not factor through U, then S' is not a strong generating set for G, hence neither is S. This justifies returning `FALSE` at this point. Otherwise, if Step 3 succeeds, then we know that S' generates G. Thus if Steps 1-3 succeed, then G is a path monoid with associated 5-tuple (S', n, h_G, U^*, f_G). Moreover, we know that (R_1) is satisfied because each $g \in S'$ factors through U, and (R_2) and (R_3) are satisfied since U^* arises as the set of path products for a labelled branching. Thus, by Theorem 4.1, it suffices to test if each basic generator factors through U and this is precisely what is accomplished in Step 4.

For the purpose of computing the running time, we will assume that the Strong Generating Test succeeds. Step 1 takes time $O(|S|n)$ by Lemma 2.5. Furthermore, if this step succeeds, then we know that $|S'| \leq \min(m\log(n), n-1)$. It takes $O(n^2)$ time to construct $\mathcal{B}$ using Build-Branching by Theorem 2.1(iii). Since $\mathcal{B}$ has m internal nodes, each call to Procedure Factor using the labelled branching data structure to store U takes time $O(mn)$. Thus Step 3 takes time $O(mn|S|)$. Finally, by Proposition 3.5, there are at most $\min(m\log(n), n-1)n$ basic generators. Thus testing, in Step 4, whether each factors through U takes time $O(\min(m\log(n), n-1)mn^2)$. The result now follows by accumulating the worst case times for each step. $\square$

Corollary 4.3. *Let G be a permutation group on an n-element set and let S be set of generating permutations for G. Then it is possible to test in time $O(|S|n + n^4)$ if S is a strong generating set.*

Remarks.

(i) Theorem 4.2 has been used to give an algorithm for completing a labelled branching for a permutation group (Cooperman *et al.* 1989), which in computer experiments, runs substantially faster than Jerrum's original algorithm (Jerrum 1986).

(ii) The set of basic generators for the the path monoid G with 5-tuple (S, n, h_G, U^*, f_G) can be identified with a subset of the Schreier generators formed from the point stabilizer sequence in a natural way. For fixed i, if $p_{ij} \in U^{(i)}, i < j$ and $g \in S^{(i)}$ such that $p_{ij}g$ is a basic generator, then $p_{ij}g$ corresponds to the Schreier generator $p_{ij}gp^{-1}$ where $p \in U^{(i)}$ is the unique element which moves i to $i^{p_{ij}g}$.

Next, we apply Theorem 3.1 to the construction of presentations. Let G be a permutation group on an n-element set and let S be strong generating set for G and let U be a complete family of coset representatives for the point stabilizer sequence. G is a path monoid as described in Example 1 of section 3, with associated 5-tuple (S, n, h_G, U^*, f_G). Let Γ be a finite set, with one-one and onto map, $\eta: \Gamma \to S$ and let H be the free monoid on Γ. Then G induces a path monoid on H, with associated 5-tuple, $(\Gamma, n, h_H, \mathcal{U}, f_H)$, as in Example 2 of section 3. Assume that η has been extended to an epimorphism from H onto G and that η' is given as in Example 2. Then it was shown that if G is properly factorizable under $=$, then H is properly factorizable under the congruence relation $\equiv_{\mathcal{H}}$ defined by $x \equiv_{\mathcal{H}} y$ if $\eta(x) = \eta(y)$. In this case, η' is defined on all of G and $\eta \circ \eta'$ is the identity on G.

Define $w_H \colon H \to H$ by $w_H = \eta' \circ \eta$. Then, as shown in Example 2, $x \equiv_{\mathcal{H}} w_H(x)$ and $w_H(x)$ is in factored form. Let $\mathcal{R}$ be the following set of equations on H with $\mathcal{T}$ the set of basic generators.

($\mathcal{R}_1$) $\forall \gamma \in \Gamma,\ \gamma = w_H(\gamma)$

($\mathcal{R}_2$) $\forall \rho_{ij}, \rho_{jk} \in \mathcal{U},\ \rho_{ij}\rho_{jk} = w_H(\rho_{ik})$

($\mathcal{R}_3$) $\forall \rho_{ik}, \rho_{jk} \in \mathcal{U},\ i < j,\ \rho_{ik} = w_H(\rho_{ij}\rho_{jk})$

($\mathcal{R}_4$) $\forall \tau \in \mathcal{T},\ \tau = w_H(\tau)$

Theorem 4.4. *A presentation for G is given by generators Γ and relations $\mathcal{R}$.*

Proof. By the definition of η and $\equiv_{\mathcal{H}}$ in Example 2 of section 3, η can be used to define an isomorphism

$$\overline{\eta} \colon H/{\equiv_{\mathcal{H}}} \to G.$$

In particular, each $\equiv_{\mathcal{H}}$-class of H contains a unique factored word of H.

Let $\equiv_{\mathcal{R}}$ be the congruence relation on H defined by closure under $\mathcal{R}$ and the substitution axiom. We will use Theorem 3.1 to prove that $\equiv_{\mathcal{H}}$ and $\equiv_{\mathcal{R}}$ are the same. First observe that $\equiv_{\mathcal{H}}$ is a refinement of $\equiv_{\mathcal{R}}$, since $\eta(x) = \eta(y)$ for each equation $x = y$ given by $\mathcal{R}_1$-$\mathcal{R}_4$. In order to prove equality, it suffices to show that $|H/{\equiv_{\mathcal{R}}}| \le |H/{\equiv_{\mathcal{H}}}|$. This in turn will follow directly once we have shown that H is properly factorizable under $\equiv_{\mathcal{R}}$.

If $p_{ij}g \in G$ is a basic generator of type T_1, then $g \in \langle S^{(i+1)} \rangle$ and $p_{ij}g$ moves i to j. Since U is a complete family of coset representatives, G is properly factorizable, and $p_{ij}g = wp_{ij}$, where $w \in G^{(i+1)}$ is properly factorizable. Therefore, $\eta'(p_{ij}g) = \eta'(wp_{ij}) = \omega\rho_{ij}$, where $\omega \in H^{(i+1)}$ is in factored form. So, $\mathcal{R}_4$ implies

($\mathcal{R}_5$) $\rho_{ij}\gamma \equiv_{\mathcal{R}} \omega'\rho_{ij} \in H^{(i)}$ where $\rho_{ij}\gamma$ is a basic generator of type T_1 and $\omega' \in H^{(i+1)}$ is in factored form.

Since $\mathcal{R}_1$-$\mathcal{R}_5$ correspond to conditions (i)-(v) of Theorem 3.1, it follows that H is properly factorizable under $\equiv_{\mathcal{R}}$. Hence, $\equiv_{\mathcal{H}} = \equiv_{\mathcal{R}}$.

We conclude that $H/{\equiv_{\mathcal{R}}}$ is isomorphic to G. Since $H/{\equiv_{\mathcal{H}}}$ is a group, it follows directly that $H/{\equiv_{\mathcal{R}}}$ is isomorphic to the finitely presented group given by generators Γ and relations $\mathcal{R}$. This completes the proof. $\square$

Corollary 4.5. *If G has a base of size m, then G has a presentation at most $\min(n-1, m\log(n))$ generators and $\min((n-1)^2, (n-1)m\log(n))$ relations.*

Proof. Let $\mathcal{B}$ be a complete labelled branching for G. By Theorem 2.1(i), there is a subset S of the set of edge labels of $\mathcal{B}$ which is a strong generating set for G with $|S| \le \min(n-1, m\log(n))$. If we use S for our generating set for G and $\mathcal{B}$ to define U, then $\mathcal{R}_1$–$\mathcal{R}_3$ are trivially true under equality in H. Thus by Theorem 4.4, there is a presentation for G with at most $|S|$ generators and with $|T|$ relations where T is the number of basic generators for G viewed as a path monoid with 5-tuple (S, n, h_G, U^*, f_G). But $|T| \le (n-1)|S|$ by Proposition 3.5 and the result follows. $\square$

Babai *et al.* (1988) give a new group membership algorithm with worst case running time of $O(n^4 \log^c(n))$. As an outgrowth of their work, they are able to show that any permutation group on n-elements has a presentation with $O(n^2 \log^c(n))$ relations. Corollary 4.5 represents an improvement in their result.

5. A Space-Efficient Strong Generating Test.

In this section, we specialize the strong generating test to the case where the generating set S for G has small cardinality in comparison to the degree n. In particular, if

$$m = |\{i\colon\ S^{(i)} - S^{(i+1)} \neq \emptyset\}|,$$

then $m \ll n$.

Let $(\mathcal{G}, \mu)$ be a group membership data structure for G which is fully augmented for S and let U be the family of coset representatives for the point stabilizer sequence of G used to construct $\mathcal{G}$. As discussed in section 4, in viewing G as a path-monoid with 5-tuple (S, n, h_G, U^*, f_G), an efficient implementation of the strong generating test, requires that property (P) be satisfied by U^*.

(P) $p_{ij}p_{jk} = p_{ik},\ \forall p_{ij}, p_{jk} \in U^*$

If (P) holds, then conditions (R_2) and (R_3) of section 4, will always be true. If $|S|$ is small, in comparison to n, then the Schreier vector data structure is an attractive alternative to a labelled branching from the point of view of saving space. However, (P) is unlikely to hold in this case, unless certain modifications are made. Thus our first step is to describe a new group membership data structure $(\mathcal{G}, \mu)$ for which (P) holds and which requires only $O(mn)$ storage, in comparison to $O(n^2)$ storage for a labelled branching.

Let $\mathcal{S}$ be Schreier vector data structure for G fully augmented for S and let *parent* be the array which defines the graph structure for a labelled branching for G fully augmented for S. $\mathcal{S}$ and *parent* can be built using Augment and Build-Schreier-Vector, respectively. Denote by σ_{ij} an element of $\langle S^{(i)} \rangle$ computed from $Schreier(i)$ which moves i to j, where $j \in i^{\langle S^{(i)} \rangle}$ and $i < j$. Let τ be the following array. Set $\tau[i] = \texttt{NIL}$ for each leaf node i. For each root node r, set $\tau[r]$ to the identity and then by induction, for each interior node j which is not a root, set $\tau[j] = \tau[i]\sigma_{ij}$ where $i = parent[j]$. Note that τ requires only mn storage as opposed to n^2 storage required for the labelled branching $\mathcal{B}$ defined in section 2.

We define the group membership data structure $(\mathcal{G}, \mu)$ by setting $\mathcal{G} = (\mathcal{S}, parent, \tau)$ and using the following algorithm to evaluate $\mu(i,j)$ for $i < j$.

```
If $j$ is a descendant of $i$ then
    If $j$ is an interior node then return($\tau[i]^{-1}\tau[j]$)
    Else let $k = parent[j]$
        If $i = k$ then return($\sigma_{kj}$)
        Else return($\tau[i]^{-1}\tau[k]\sigma_{kj}$)
Else return(NIL)
```

If the cost of computing a coset representative directly from the Schreier data structure is c, then the cost based on this hybrid structure will be $c + O(n)$. If j is a leaf node, the cost will usually be dominated by the original cost c.

The following result is a direct consequence of Theorem 2.1 and the fact that the computation of each σ_{ij} using $\mathcal{S}$ takes time $O(n^2)$.

Lemma 5.1. *It takes $O(mn^2 + |S|n)$ time to construct $(\mathcal{G}, \mu)$ and $O(n^2)$ time to evaluate $\mu(i,j)$.*

It is trivial to show that if U is the family of coset representatives for the point stabilizer sequence defined by $(\mathcal{G}, \mu)$ and if G is viewed as a path monoid with associated 5-tuple