Applications of Computer Algebra (ACA-2004)

Raid Braid: Fast Conjugacy Disassembly in Braid and Other Groups

Patrick D. Bangert

Abstract. The conjugacy problem in finitely presented groups is an important old problem that has gained prominence recently via group theoretical encryption methods. The theory of rewriting systems is extended to allow conjugacy problems in certain groups to be solved using rewriting methods. We construct explicit rewrite systems to solve the word and conjugacy problems in the braid groups. We show that the complexity of both algorithms is polynomial-time. This resolves an old problem in low-dimensional topology and shows that the braid group is not a suitable platform group for cryptography, against expectations in the literature.

1. Introduction

The braid group can be defined by its Artin presentation,

$$B_n = \langle \{\sigma_i\} : \ \sigma_i\sigma_j \approx \sigma_j\sigma_i, \ \ \sigma_i\sigma_{i+1}\sigma_i \approx \sigma_{i+1}\sigma_i\sigma_{i+1}, \ \ i,j < n, \ \ |i-j| > 1 \rangle \tag{1}$$

There is a popular geometric interpretation of the generators σ_i of this group in which there are n strings all of which are extended vertically upwards with the exception of the i^{th} and $i+1^{st}$ string that exchange places such that the i^{th} string is closer to the observer; the order of these strings reverses in the inverse of the generator σ_i^{-1}. In what follows we shall use the symbol $\approx$ to mean that two words in the above generators are equivalent in the sense that one may be transformed into the other by the relations in B_n and we use the $=$ sign to mean letter by letter equality. Furthermore, we denote the length of the braid word a by $L(a)$ and the number of strings (the braid group it belongs to) by $n(a)$.

We may ask the word, conjugacy and Markov problems as follows: Given two braids $a, b \in B_n$, the *word problem* asks whether $a \approx b$ and the *conjugacy problem* asks whether there exists a word $c \in B_n$ such that $a \approx cbc^{-1}$ (we denote conjugacy by $\approx_c$). For two braids $a \in B_n$ and $b \in B_m$, the *Markov problem* asks whether it is possible to get from a to b using a finite number of conjugations and stabilizations (a stabilization is the move $a \to a\sigma_n^{\pm 1}$ or its inverse). It is clear that the word problem is a special case of the conjugacy problem which is a special case of the Markov problem.

School of Engineering and Science, International University Bremen, P.O. Box 750 561, 28725 Bremen, Germany; p.bangert@iu-bremen.de

There is much cause to study these problems for many reasons. Alexander proved that any knot can be represented as a braid provided that the braid is closed (its endpoints at both the top and bottom are pairwise identified) [**1**]. Subsequently, Markov claimed and Birman proved that any two closed braids (i.e. knots) are ambient isotopic if and only if they are Markov equivalent [**7**]. Thus, the clearly important problem of knot classification becomes a problem in combinatorial group theory.

Thus far, no solution to the Markov problem is known. As the word and conjugacy problems are sub-problems of it, they can be attacked first and have both been solved many times. The word problem was first solved by Artin in the same paper that first introduced braids in 1925 [**3, 4**] whereas the conjugacy problem was first solved by Garside in 1969 [**14**]. The best solutions known to date are presented in [**8, 18**].

The paper is organized as follows. We review the major current application of our result in the theory of cryptography. A word problem solution using rewrite systems is constructed next. We then extend the theory of rewrite systems (mainly Newman's Lemma and the Critical Pair Lemma) and Knuth-Bendix completion to the case of circular words. This extension is used to extend the word problem solution to a conjugacy problem solution. Finally the correctness and complexity of the solution is analyzed. We assume throughout that the reader is familiar with the concepts of term rewriting systems; a suitable introduction may be found in [**5**].

2. Group-theoretic Cryptography

There are many practical applications of algorithms that solve either one of these problems. A recently prominent application is in group theoretical cryptography in which the encryption keys are group elements [**2**]. This public-key system takes a platform group in which the two corresponding partners have to solve word problems in order to decrypt their messages. The intruder, who lacks the private keys, needs to solve a conjugacy problem to break the cypher. Thus we are looking for a group in which we have a fast solution for the word problem and a very difficult conjugacy problem. It has been proposed that the braid groups are suitable because the most efficient word problem algorithm runs in $O(L(a)^2 n(a))$ time [**8**] whereas *all* known conjugacy algorithms run in exponential time in both $L(a)$ and $n(a)$. All attempts at proving the conjugacy problem to be difficult (such as NP-complete for instance) have not met with success. This paper constructs a polynomial-time algorithm for the conjugacy problem and thus shows that a different group must be sought. It also raises afresh the question of an algorithmic solution to the Markov problem (i.e. combinatorial knot classification) that has virtually been abandoned due to the supposed difficulty of the conjugacy problem.

There have been many attempts to break the group-theoretic cypher with the braid groups as platform, as proposed by Anshel *et. al.* [**2**]. It is plain that a high worst-case complexity for the conjugacy problem does not yet save the platform group as the average-case complexity could be exponentially lower. This is the basis for probability attacks in which the algorithm is polynomial-time but not guaranteed to succeed [**15, 21**]. An algorithm for the conjugacy problem in the braid groups has been devised that is linear in the size of the super-summit set [**12**] of a braid; while no concrete size estimates are available, it seems clear that the

size is exponentially growing with length and group index [**13, 16**]. One may try to solve the conjugacy problem in other presentations but this does not appear to reduce the complexity [**20**]. A (strictly) sub-problem of the conjugacy problem has been solved in polynomial-time [**9**]: The Diffie-Hellman conjugacy problem that asks to find $baua^{-1}b^{-1}$ for given u, aua^{-1}, bub^{-1} for a and b in two commuting subgroups of B_n.

3. The Word Problem in B_n

To solve the word problem, we begin with the braid group and form an associated monoid by adding the relation defining the inverse generators and the definition of the generator of the center of the braid groups, $\Delta_n^2 = (\sigma_1\sigma_2\cdots\sigma_{n-1})^n$ [**10**].

$$\begin{aligned} M^+(B_n) = \langle \{\sigma_1^{\pm 1}, \sigma_2^{\pm 1}, \cdots, \sigma_{n-1}^{\pm 1}, \Delta_n^{\pm 2}\} : \Delta_n^{\pm 2}\sigma_i = \sigma_i\Delta_n^{\pm 2}; \\ \Delta_n^{\pm 2}\Delta_n^{\mp 2} = \sigma_i^{\pm 1}\sigma_i^{\mp 1} = e; \\ \sigma_i^{\pm 1}\sigma_j^{\pm 1/\mp 1} = \sigma_j^{\pm 1/\mp 1}\sigma_i^{\pm 1} \text{ for } |i-j| > 1; \\ \sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1} = \sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1} \rangle \end{aligned} \tag{2}$$

This choice will allow a complete rewriting system to be deduced. It should be clear that if we are successful in solving the word and conjugacy problems in $M^+(B_n)$, the problems in B_n are solved. For convenience, we define

$$a_{i,j} = \sigma_i\sigma_{i+1}\cdots\sigma_j \tag{3}$$

$$d_{i,j} = \sigma_i\sigma_{i-1}\cdots\sigma_j \tag{4}$$

We now define a total order $<_b$ on the letters of our alphabet

$$\Delta_n^2 <_b \Delta_n^{-2} <_b \sigma_1 <_b \sigma_2 <_b \cdots <_b \sigma_{n-1} <_b \sigma_1^{-1} <_b \sigma_2^{-1} <_b \cdots <_b \sigma_{n-1}^{-1} \tag{5}$$

Having formulated the problem thus, we use Knuth-Bendix completion to obtain our rewrite rules. In practice, this process is laborious and would occupy prodigious space if described in detail. For this reason, we will simply state the result and prove it to be correct.

For what follows, we shall represent a braid of the form $\Delta_n^{2k}P$ as the pair (k, P). The reason for this is to effectively remove any sub-braid which lies in the center of the braid group B_n from the braid in question. The reason for this will become apparent when we extend our solution to the conjugacy problem. Removing any Δ_n^{2k} from any part of a braid, while remembering how many have been so removed, can be done without loss of information because Δ_n^2 is the generator of the center of B_n and thus its position is irrelevant. We obtain the following rewriting system.

$$\begin{aligned} \mathcal{W}_n = \{ &(1)\ \sigma_i^{-1} \to \prod_{j=1}^{i-1}[d_{j,1}a_{1,j}]\, d_{i,1}a_{1,i-1} \prod_{j=i+1}^{n-1}[d_{j,1}a_{1,j}] \ \ \&\ \ k \to k-1; \\ &(2)\ \sigma_i\sigma_j \to \sigma_j\sigma_i \text{ for } j < i-1; \\ &(3)\ \sigma_i\sigma_{i-1}P\sigma_i \to \sigma_{i-1}\sigma_i\sigma_{i-1}P; \\ &(4)\ \sigma_i\sigma_{i-1}Q\sigma_{i-1}Rd_{i,j} \to \sigma_{i-1}\sigma_i\sigma_{i-1}Qd_{i-1,j}\sigma_i R^+ \text{ for } j < i; \\ &(5)\ \prod_{i=1}^{n-1} d_{i,1}a_{1,i}S_i \to \prod_{i=1}^{n-1} S_i \ \ \&\ \ k \to k+1\ \} \end{aligned} \tag{6}$$

The variables P, Q, R and S_i are (possibly empty) words in the generators σ_k (and *not* their inverses σ_k^{-1}) subject to the restriction that the highest generator index k is $i-2$, $i-2$, $i-1$ and i respectively and the lowest generator index in R is j, where i and j refer to the values of the generator indices of the respective rules. The word R^+ is obtained from R by increasing all generator indices in R by one. Note that rules 1 and 5 require two replacements to be made simultaneously. Rules 1 and 5 are simple to understand; the other rules are illustrated in figure 1.

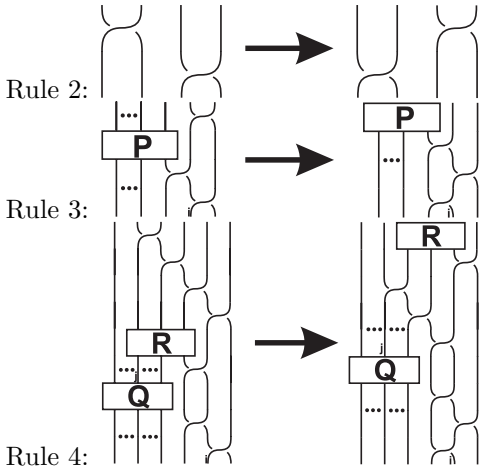

FIGURE 1. Rules 2, 3 and 4 of TRS $\mathcal{W}_n$ illustrated.

THEOREM 3.1. *$\mathcal{W}_n$ is complete and thus solves the word problem for B_n.*

Proof. *(termination)* Every application of $\mathcal{W}_n$ simplifies any word with respect to $<_b$. As $<_b$ is well-founded, $\mathcal{W}_n$ terminates.

(local confluence) There are 20 overlaps between the rules in $\mathcal{W}_n$ and none give rise to a critical pair. For reasons of space, we do not provide all the reduction steps for each overlap but list all overlaps, the rules from which they arise and the common reduct of all reduction paths of the overlap in table 1. The restrictions on the indices and the variables are obvious from the context and the definition of $\mathcal{W}_n$. The dedicated reader may easily but laboriously verify that the list is both complete and correct. There are an additional 16 (four variables and four positive redexes) variable overlaps, i.e. overlaps in which a variable completely contains a redex, but these resolve trivially and so are not listed in the table. By the Critical Pair Lemma, we thus conclude that the system is locally confluent.

(confluence) As the system terminates and is locally confluent, Newman's Lemma allows us to conclude that the system is confluent and thus complete.

(equivalence) We now need to demonstrate that when the arrows are replaced by equal signs, we retrieve the monoid's relations exactly. Rules 2 and 3 imply both braid group relations. Rule 5 represents the definition of Δ_n^2 in terms of the σ_i. The second parts of rules 1 and 5 imply that Δ_n^2 and Δ_n^{-2} are inverses. Rule 1, after use of the braid group relations, the definition of Δ_n^2 and Δ_n^{-2} indicates that σ_i and σ_i^{-1} are inverses. All (and only) the relations in the monoid $M^+(B_n)$ are thus contained in $\mathcal{W}_n$. Birkhoff's theorem thus tells us that this system is equivalent to

the monoid $M^+(B_n)$. As it is complete as well, it thus solves the word problem by reducing each word to its normal form. □

TABLE 1. Overlaps between rules in $\mathcal{W}_n$. (The ellipses, $\cdots$, indicate line breaks and not pattern continuation signs.)

$\rightarrow$	Overlap	Final Form
2, 2	$\sigma_i\sigma_j\sigma_k$	$\sigma_k\sigma_j\sigma_i$
2, 3	$\sigma_i\sigma_j\sigma_{j-1}P\sigma_j$	$\sigma_{j-1}\sigma_j\sigma_{j-1}P\sigma_i$
2, 3	$\sigma_i\sigma_{i-1}\sigma_jP\sigma_i$	$\sigma_j\sigma_{i-1}\sigma_i\sigma_{i-1}P$
2, 3	$\sigma_i\sigma_{i-1}P\sigma_i\sigma_j$	$\sigma_{i-1}\sigma_i\sigma_{i-1}P\sigma_j$
2, 4	$\sigma_i\sigma_j\sigma_{j-1}Q\sigma_{j-1}Rd_{j,k}$	$\sigma_{j-1}\sigma_j\sigma_{j-1}Qd_{j-1,k}\sigma_jR^+\sigma_i$
2, 4	$\sigma_i\sigma_{i-1}\sigma_kQ\sigma_{i-1}Rd_{i,j}$	$\sigma_k\sigma_{i-1}\sigma_i\sigma_{i-1}Qd_{i-1,j}\sigma_iR^+$
2, 4	$\sigma_i\sigma_{i-1}Q\sigma_{i-1}Rd_{i,j}\sigma_k$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Q\sigma_kd_{i-1,j}\sigma_iR^+$
2, 5	$\prod_{i=1}^{n-1} d_{i,1}a_{1,i}S_i; S_j = \sigma_kS'_j$	$\prod_{i=1}^{n-1} S_i; S_j = \sigma_jS'_j$
3, 3	$\sigma_i\sigma_{i-1}P\sigma_i\sigma_{i-1}P'\sigma_i$	$\sigma_{i-1}\sigma_i\sigma_{i-1}P\sigma_{i-1}P'\sigma_i$
3, 4	$\sigma_i\sigma_{i-1}P\sigma_i\sigma_{i-1}Q\sigma_{i-1}Rd_{i,j}$	$\sigma_{i-1}\sigma_i\sigma_{i-1}P\sigma_{i-1}Q\sigma_{i-1}Rd_{i,j}$
3, 4	$\sigma_i\sigma_{i-1}\sigma_{i-2}P\sigma_{i-1}R\cdots$ $\cdots\sigma_{i-2}R'\sigma_{i-1}R''d_{i,j}$	$\sigma_{i-2}\sigma_{i-1}\sigma_{i-2}\sigma_i\sigma_{i-1}\sigma_{i-2}P\cdots$ $\cdots Rd_{i-2,j}\sigma_{i-1}R'^+\sigma_iR''^+$
3, 4	$\sigma_i\sigma_{i-1}\sigma_{i-2}P\sigma_{i-1}R\sigma_{i-1}R'd_{i,j}$	$\sigma_{i-2}\sigma_{i-1}\sigma_{i-2}\sigma_i\sigma_{i-1}\sigma_{i-2}PRd_{i-2,j}\sigma_iR'^+$
3, 4	$\sigma_i\sigma_{i-1}\sigma_{i-2}P\sigma_{i-1}R\sigma_{i-2}R'd_{i,j}$	$\sigma_{i-2}\sigma_{i-1}\sigma_{i-2}\sigma_i\sigma_{i-1}\sigma_{i-2}PRd_{i-2,j}\sigma_{i-1}R'^+$
3, 4	$\sigma_i\sigma_{i-1}\sigma_{i-2}P\sigma_{i-1}Rd_{i,j}$	$\sigma_{i-2}\sigma_{i-1}\sigma_{i-2}\sigma_i\sigma_{i-1}\sigma_{i-2}PRd_{i-2,j}$
3, 4	$\sigma_i\sigma_{i-1}P\sigma_{i-1}Rd_{i,j}R'\sigma_k$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Pd_{i-1,j}\sigma_iR^+R'\sigma_k$
3, 5	$\prod_{i=1}^{n-1} d_{i,1}a_{1,i}S_i; S_j = \sigma_{j-1}P\sigma_jS'_j$	$\prod_{i=1}^{n-1} S_i; S_j = \sigma_{j-1}P\sigma_jS'_j$
4, 4	$\sigma_i\sigma_{i-1}Q\sigma_{i-1}Rd_{i,j}Q'\sigma_{k-1}R'd_{k,m}$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Qd_{i-1,j}\sigma_iR^+Q'\sigma_{k-1}R'd_{k,m}$
4, 4	$\sigma_i\sigma_{i-1}\sigma_{i-2}Q\sigma_{i-2}Rd_{i-1,j}Q'd_{i,k}$	$\sigma_{i-2}\sigma_{i-1}\sigma_{i-2}\sigma_i\sigma_{i-1}\sigma_{i-2}Q\cdots$ $\cdots d_{i-2,k}d_{i-1,j+1}\sigma_iR^{++}Q'^+$
4, 4	$\sigma_i\sigma_{i-1}Q\sigma_{i-1}Rd_{i,j}$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Qd_{i,j}\sigma_iR^+$
4, 5	$\prod_{i=1}^{n-1} d_{i,1}a_{1,i}S_i$ $S_j = \sigma_{j-1}Q\sigma_{j-1}Rd_{j,k}S'_j$	$\prod_{i=1}^{n-1} S_i$ $S_j = \sigma_{j-1}Q\sigma_{j-1}Rd_{j,k}S'_j$

The rules of a rewrite system are to be applied in a non-deterministic way and a complete rewrite system always reaches the unique normal form no matter what strategy of rule application is chosen [**5**]. Algorithm 3.2 presents one such strategy the complexity of which in proven in theorem 3.3.

ALGORITHM 3.2. *Input: A word $w \in B_n$. Output: A word $w' \in B_n$ which is the unique representative of the equivalence class of w.*

(1) Apply rule 1 of $\mathcal{W}_n$ as many times as possible.
(2) Apply rules 2, 3, 4 and 5 of $\mathcal{W}_n$ as many times as possible in order proceeding to the next rule only if the current can no longer be applied.
(3) Loop step 2 until no rule may be applied to the word at all. In this case w' has been found.

THEOREM 3.3. *$\mathcal{W}_n$ solves the word problem for any word $w \in B_n$ with complexity $O\left(L(w)^2 n(w)^4\right)$.*

Proof. Suppose that w contains exactly m inverse generators. Rule 1 may be applied exactly m times; note that m goes as $O(L(w))$. We must search the word for the redexes of rule 1 and then replace them. Searching is an $O(L(w))$ operation but the reducts increase in length as $O(n(w)^2)$ and thus the application of rule 1 takes time $O(L(w)n(w)^2)$. It is clear that rule 1 may never be applied again and the word length of w is now $L_2(w) = L(w) + mn(w)(n(w)-1) - m = O\left(L(w)n(w)^2\right)$. Rule 2 may be applied a number of times bounded by $L_2(w)^2$ as it is a pairwise comparison between all generators in the word at worst. An application of rules 3 and 4 may give rise to a further application of itself or the other rule but strictly later in the word and thus the number of times they may be applied is bounded by $L_2(w)$. While rules 2, 3 and 4 keep the word length constant, rule 5 reduces it by $n(w)(n(w)+1)$ and thus rule 5 may be applied a maximum of

$$\frac{L(w) + mn(w)(n(w)-1) - m}{n(w)(n(w)+1)} = \frac{O\left(L(w)n(w)^2\right)}{O\left(n(w)^2\right)} = O\left(L(w)\right) \tag{7}$$

times. Thus the total worst-case complexity of the algorithm is $O(L_2(w)^2) = O(L(w)^2 n(w)^4)$. The application of rule 2 is responsible for the quadratic behavior in the length; it is the bottleneck of the calculation. □

We have already noted that the most efficient algorithm known has complexity $O(L(w)^2 n(w))$ and so our algorithm is slower by a factor of $n(w)^3$ [**8**]. The purpose of presenting it here is (1) that this algorithm is very easy to understand and apply as it is just a set of five rewrite rules (past algorithms were much more involved) and (2) that this algorithm generalizes to the conjugacy case.

4. Rewriting Systems for Conjugacy Problems

We wish to extend the rewriting system above to the case of conjugacy and, as some labor will show, this is not possible using existing methods. Thus, the theory of rewriting systems must be slightly extended to deal with this case and this is what we shall do in this section. It will be apparent that these methods are applicable to a wide range of groups and is certainly not restricted to the braid groups.

We first describe the idea and then give the details of the method. We find that the notion of completeness generalizes to the new setting and then proceed to prove that the conditions for completeness also generalize. This leaves us with the construction of two methods; namely the checking for termination and critical pairs. For the new setting such methods are given and proved correct. All we now have to do in a particular setting is check these two conditions; if they hold we have a valid conjugacy problem solution. At the end of the paper we actually check them for the braid group and find them to hold.

4.1. The Idea for Free Groups. First, we shall illustrate the idea behind the extension and then work out the details. Suppose that $G = \mathcal{F}_n$ the free group of rank n. This group is generated by n elements $\{f_i\}$ for $1 \leq i \leq n$ and no relations [**19**]. A general word $w \in \mathcal{F}_n$ takes the form

$$w = f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_m}^{p_m}, \qquad 1 \leq s_k \leq n \tag{8}$$

Since there are no relations in $\mathcal{F}_n$, the word w is unique over its equivalence class if and only if $s_i \neq s_{i+1}$ for all i. This condition is trivially obtained from any word $w \in \mathcal{F}_n$ by applying the (obviously) complete rewriting system

$$\mathcal{R}_w(\mathcal{F}_n) = \{f_s^p f_s^q \to f_s^{p+q},\ \forall 1 \leq s \leq n\} \tag{9}$$

Thus $\mathcal{R}_w(\mathcal{F}_n)$ solves the word problem in any free group $\mathcal{F}_n$. Moreover, it does so in a time proportional to $L(w)$.

Consider now the conjugacy problem in $\mathcal{F}_n$. We define the i^{th} *cyclic permutation* $C^i(w)$ of a word w in the general form of equation (8) by

$$C^i(w) = f_{s_j}^{p_j'} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_j}^{p_j''} \tag{10}$$

such that

$$p_j' + \sum_{k=j+1}^{m} p_k = i \tag{11}$$

Intuitively, the i^{th} cyclic permutation is obtained by taking the last i generators in the word w and moving them to the front of the word one by one.

Definition 4.1. *Two words w and w' are* cyclicly permutable *(denoted $\approx_{cp}$) if and only if there exists a finite sequence of moves from w to w' where each move is a cyclic permutation or a word equivalence move in the group.*

Remark 4.2. *Let us consider an alphabet of three letters a, b and c with the equation $ab = bc$. Then $abc \approx_{cp} bca$ as the right-hand side is the second cyclic permutation of the left-hand side. We also have that $abc \approx_{cp} bcc$ by not permuting at all but applying a word equivalence. Finally, it is obvious that cyclic permutability forms an equivalence relation for any group G. We give this example to make the definition clear lest it cause confusion in connection with the statement that the equivalence relation of cyclic permutability is the same as that of conjugacy (see the proposition below). Conjugating the word w with the word q is nothing more than appending qq^{-1} to the end of w and then moving the last $L(q)$ letters to the front of the word. Thus this proposition should be clear now.*

Proposition 4.3. *For any group G and any two words $w, w' \in G$, we have $w \approx_{cp} w'$ if and only if $w \approx_c w'$.*

Proof. Any group G has a presentation which may be obtained from some free group $\mathcal{F}_n$ of rank n by adding relations [**11**]. Moreover, if the conjugacy problem is solvable in one representation, it is solvable in all [**22**]. Suppose $w \approx_{cp} w'$, then there exists an i for which

$$w' \approx C^i(w) = f_{s_j}^{p_j'} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_j}^{p_j''} \tag{12}$$

where

$$p_j' + \sum_{k=j+1}^{m} p_k = i \tag{13}$$

Let

$$\gamma = f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_j}^{p_j''} \tag{14}$$

Then

$$w' \approx f_{s_j}^{p_j'} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} \gamma \tag{15}$$

$$\approx \gamma^{-1} \gamma f_{s_j}^{p_j'} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} \gamma \tag{16}$$

$$\approx \gamma^{-1} w \gamma \tag{17}$$

Thus we have $w \approx_c w'$. Now suppose $w \approx_c w'$, then there exists a γ such that

$$w' \approx \gamma^{-1} w \gamma \tag{18}$$

If the word length of γ is $L(\gamma)$, then we have

$$C^{L(\gamma)}(w') \approx \gamma\gamma^{-1} w \approx w \tag{19}$$

Thus $w \approx_{cp} w'$. □

The central problem in generalizing a word problem solution to a conjugacy problem solution lies in the fact that any letter in the word can be viewed as the first letter in the conjugacy case. One has to find a convenient way to encapsulate this extra freedom. This is the underlying idea that gave rise to the above definition and the subsequent development. Pictorially this can be viewed as no longer writing a word linearly but circularly; it being understood that the order of the letters is preserved and only the identity of the first letter is lost (see figure 2).

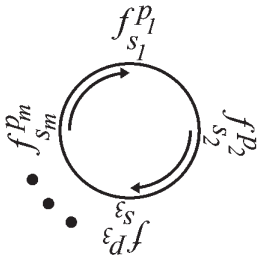

FIGURE 2. The word w given in equation (8) bent into a circle. While the circularity removes the notions of beginning and end of a word, it preserves the directionality of it.

DEFINITION 4.4. *From a word w, we form the* cyclic word $c(w)$ *that is a set having as its members $C^i(w)$ for all $0 \leq i < L(w)$.*

We further define that a rewrite rule is *applicable* to a cyclic word $c(w)$ if it is applicable normally to at least one of its members, $w' \in c(w)$ say. If a rewrite rule is applicable, let the member w' of the cyclic word $c(w)$ be rewritten in the standard way and then let $c(w)$ be replaced with $c(w')$ where w' is understood to have been rewritten. This method of letting a rewrite system act on a cyclic word allows us to forget about conjugacy moves as we shall see.

A rewrite system is called *cyclicly locally confluent*, *cyclicly confluent* and *cyclicly terminating* if it is locally confluent, confluent and terminating respectively for cyclic words with the above method of application in mind. Furthermore, a rewrite system is *cyclicly complete* if it is both cyclicly confluent and cyclicly terminating. The above discussion proves the following theorem.

THEOREM 4.5. *A given cyclicly complete rewrite system solves the conjugacy problem for the group with the given alphabet as generators and the relations obtained from the rewrite rules by taking the reflexive-transitive-symmetric closure.*

This leaves us with deciding when a rewrite system is cyclicly complete. It turns out that Newman's Lemma and the Critical Pair Lemma generalize to the cyclic scenario and so: cyclic local confluence implies confluence if the system cyclicly terminates and the cyclic local confluence can be checked by the absence of critical pairs. The following two sections will be devoted to proving these results.

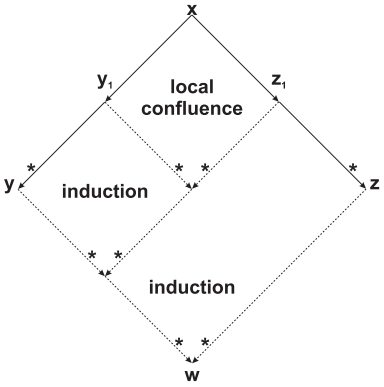

FIGURE 3. The proof of Newman's Lemma (lemma 4.6) in diagrammatic form. We begin at the top with a local divergence which is rectifiable by assumption and thus by induction any global divergence is also rectifiable. It is because of this diagrammatic proof that Newman's Lemma is also known as the Diamond Lemma.

4.2. The Cyclic Newman's Lemma.

LEMMA 4.6. *A cyclicly terminating rewrite system is cyclicly confluent if and only if it is cyclicly locally confluent.*

Proof. This proof is similar to the one given for the standard Newman Lemma in [**17**]. The result is obvious from figure 3. If $\rightarrow$ is a rewrite rule, then $\leftarrow$ is its inverse and $\rightarrow^*$ its reflexive-transitive closure.

(if) We want to show that if $y \leftarrow^* x \rightarrow^* z$, then the final forms of y and z are identical, which exist since the rewrite system cyclicly terminates. If $x = y$ or if $x = z$, the result is obvious. If $x \rightarrow y_1 \rightarrow^* y$ and $x \rightarrow z_1 \rightarrow^* z$, then there exists a u such that $y_1 \rightarrow^* u \leftarrow^* z_1$ by cyclic local confluence. The existence of a w such that $y \rightarrow^* w \leftarrow^* z$ follows by induction over arbitrary length rewriting paths; the finiteness of all rewriting paths is attested to by cyclic termination.

(only if) This it trivial as cyclic local confluence is subsumed by cyclic confluence. □

4.3. The Cyclic Critical Pair Lemma. The Critical Pair Lemma states that a rewrite system is locally confluent if and only if it has no critical pairs. Recall that a critical pair arises from an overlap of two redexes in a word which gives rise to a local divergence of rewriting paths which do not meet again. Given a rewrite system $\mathcal{R} = \{(l_i, r_i)\}$, a *cyclic overlap* is a cyclic word $c(w) = c(abcd)$ such that $abc = \rho l_i$ and $cda = \eta l_j$ for some words a, b, c and d, two (possibly equal) integers i and j and substitutions ρ and η. The cyclic overlap $c(abcd)$ is rewritten to both $c(\rho r_i d)$ and $c(b\eta r_j)$. A cyclic overlap is *non-critical* if the reducts are joinable, $c(\rho r_i d) \leftrightarrow^\star c(b\eta r_j)$ and *critical* otherwise. A *cyclic critical pair* is the (unordered) pair of cyclic words $(c(\rho r_i d), c(b\eta r_j))$ which arises from a cyclic critical overlap. It is obvious that if $\mathcal{R}$ contains cyclic critical pairs, it can not be cyclicly confluent.

For example, consider the rewrite system $\mathcal{R} = \{abxba \to cxc\}$ over the alphabet $\mathcal{A} = \{a, b, c\}$ and some variables x and y. Clearly $\mathcal{R}$ contains the cyclic critical overlap $abxbabyb$ which is to be rewritten into $bxbcyc$ and $cxcbyb$. This cyclic critical pair may be resolved by noting that if the variable contained between the c letters is less than the other, it is that cyclic word which is to be preferred under the lexicographic order $c < b < a$. That is, we have to add a conditional rule depending on the relative value of the variables. This global rule must be applied, if applicable, with preference over the ordinary local rule. In this way we have extended Knuth-Bendix completion to the cyclic case; note that all rules added in this procedure are *global* (i.e. are to be applied to the entire word) whereas the usual rules of normal rewrite systems are *local* (i.e. are to be applied to a sub-word). We shall now prove the extension of the Critical Pair Lemma for the cyclic case.

LEMMA 4.7. *A rewrite system $\mathcal{R} = \{(l_i, r_i)\}$ is cyclicly locally confluent if and only if it contains neither critical nor cyclicly critical pairs.*

Proof. We consider all relative positions of two redexes l_i and l_j in a cyclic word w and analyze them in turn. The first four cases occur in the standard Critical Pair Lemma but in the cyclic case there are five cases:

(1) *(disjoint)* Suppose $c(w) = c(l_i x l_j y)$ for some words x and y. The existence of the common reduct $c(r_i x r_j y)$ is obvious; see figure 4 (a).
(2) *(variable overlap)* Suppose l_i contains a variable which contains l_j as a sub-term. If $l_j \to r_j$ does not change the applicability of l_i, a common reduct is obvious. If it does and the divergence does not resolve, we have an instance of a critical pair; see figure 4 (b).
(3) *(critical overlap)* Suppose l_i and l_j have a critical overlap. A critical pair exists and obviously prevent local confluence; see figure 4 (c).
(4) *(orthogonal)* Suppose l_i and l_j have a non-critical overlap. By definition the divergence resolves; see figure 4 (d).
(5) *(cyclical critical overlap)* Suppose l_i and l_j have a cyclic overlap. If it is critical, we have an instance of a cyclic critical pair which obviously prevents cyclic local confluence. If it is non-critical, a common reduct exists by definition; see figure 4 (e).

□

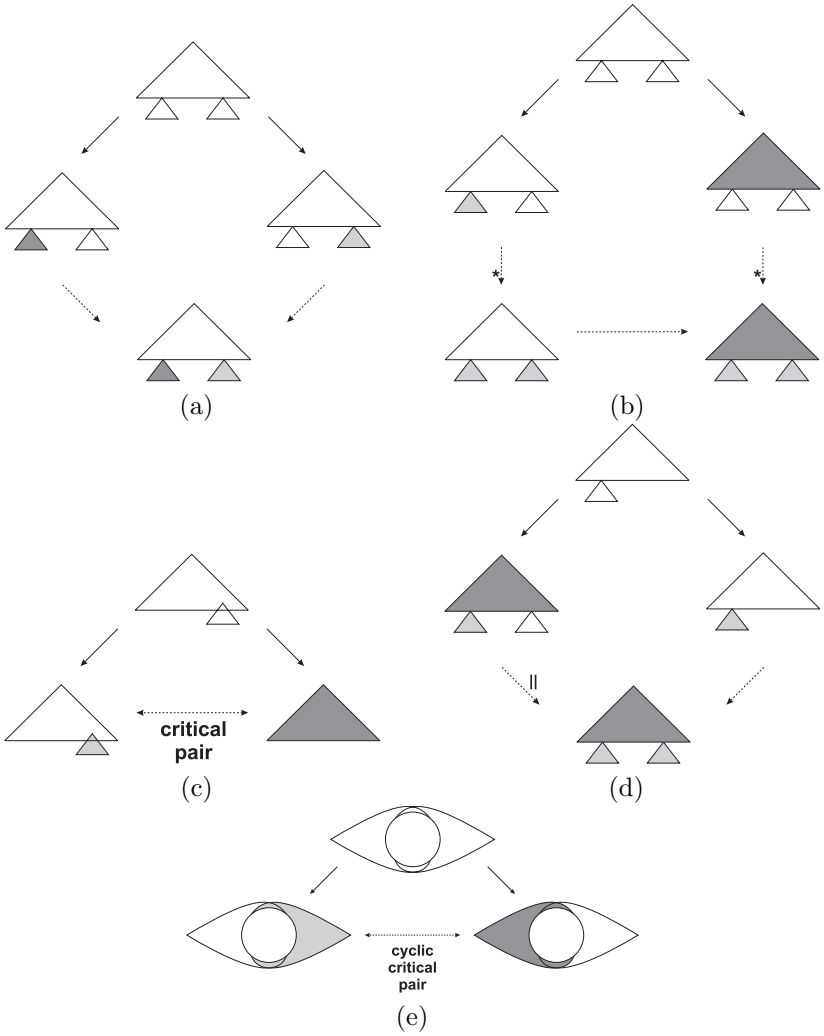

FIGURE 4. The proof of lemma 4.7 in its four cases: (a) the disjoint case, (b) the variable overlap case, (c) the critical overlap case, (d) the orthogonal case, (e) the circular critical overlap case.

5. The Conjugacy Problem in B_n

The preceding section proved that a rewrite system will solve the conjugacy problem in a group G if the reflexive-transitive-symmetric closure of its rules (the replacement of arrows with equal signs) is identical to the group's relations and provided that the rewrite system cyclicly terminates and has no critical pairs (normal or cyclic).

For the braid groups B_n, we have a working word problem solution in the form of a rewrite system. If it were cyclicly complete, we would have a conjugacy problem solution. It is not cyclicly complete but it can be made so.

TABLE 2. Cyclic overlaps between rules in $\mathcal{W}_n$.

$\rightarrow$	Cyclic Overlap	Final Forms
2, 2	$\prod_{i=1}^{n-1}(d_{i,1}a_{1,i}S_i)$	$\prod_{i=1}^{n-1} S_i$
	& $S_{n-1} = S'_{n-1}\sigma_k$; $3 \leq k < n$	$\sigma_1\sigma_k\sigma_1 S_1 \prod_{i=2}^{n-2}(d_{i,1}a_{1,i}S_i)\, d_{n-1,1}a_{1,n-1}S'_{n-1}$
3, 3	$\sigma_i\sigma_{i-1}P\sigma_i\sigma_{i-1}P'$	$\sigma_{i-1}\sigma_i\sigma_{i-1}P\sigma_{i-1}P'$
		$\sigma_{i-1}\sigma_i\sigma_{i-1}P'\sigma_{i-1}P$
3, 4	$\sigma_i\sigma_{i-1}Q\sigma_{i-1}R\sigma_i\sigma_{i-1}P$	$\sigma_{i-1}Q\sigma_{i-1}R\sigma_{i-1}\sigma_i\sigma_{i-1}P$
	& $P = d_{i-2,j}P'$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Qd_{i-1,j}\sigma_i R^+P'$
4, 4	$\sigma_i\sigma_{i-1}Q_1\sigma_{i-1}R_1\sigma_i\sigma_{i-1}Q_2\sigma_{i-1}R_2$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Q_1d_{i-1,j}\sigma_i R_1^+Q'_2\sigma_{i-1}R_2$
	& $Q_1 = d_{i-2,j}Q'_1$; $Q_2 = d_{i-2,j}Q'_2$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Q_2d_{i-1,j}\sigma_i R_2^+Q'_1\sigma_{i-1}R_1$

In table 2, we list all four cyclic overlaps between the rules of $\mathcal{W}_n$ and the two final forms per overlap depending on the chosen rewrite path. Note that all overlaps are critical and that the cyclic overlap refers to the entire cyclic word. According to our extended form of Knuth-Bendix completion, we must orient these cyclic critical pairs and add the resulting rules as global rules to the rewrite system. Consider the rewrite system $\mathcal{G}_n$ below which is understood to contain only global rules for cyclic words, i.e. the entire word has to be matched to redexes in $\mathcal{G}_n$. The restrictions on the variables are identical to those of $\mathcal{W}_n$. The ordering $<_s$ is the standard shortlex ordering, i.e. words are sorted lengthwise first and then lexicographically using $<_b$.

$$
\begin{aligned}
\mathcal{G}_n = \{\, &(1)\ \prod_{i=1}^{n-1}(d_{i,1}a_{1,i}S_i) \rightarrow \prod_{i=1}^{n-1} S_i;\\
&(2)\ \sigma_i\sigma_{i-1}P\sigma_i\sigma_{i-1}P' \rightarrow \sigma_{i-1}\sigma_i\sigma_{i-1}P\sigma_{i-1}P' \text{ if } P <_b P'\\
&\text{ or } \sigma_{i-1}\sigma_i\sigma_{i-1}P'\sigma_{i-1}P \text{ if } P' \leq_b P;\\
&(3)\ \sigma_i\sigma_{i-1}Q\sigma_{i-1}R\sigma_i\sigma_{i-1}P \rightarrow \sigma_{i-1}Q\sigma_{i-1}R\sigma_{i-1}\sigma_i\sigma_{i-1}P;\\
&(4)\ \sigma_i\sigma_{i-1}Q_1\sigma_{i-1}R_1\sigma_i\sigma_{i-1}Q_2\sigma_{i-1}R_2 \rightarrow\\
&\sigma_{i-1}\sigma_i\sigma_{i-1}Q_1d_{i-1,j}\sigma_i R_1^+Q'_2\sigma_{i-1}R_2 \text{ if } R_1 <_s R_2\\
&\text{ or } \sigma_{i-1}\sigma_i\sigma_{i-1}Q_2d_{i-1,j}\sigma_i R_2^+Q'_1\sigma_{i-1}R_1 \text{ if } R_2 \leq_s R_1\ \}
\end{aligned}
\tag{20}
$$

As described in the solution to the word problem, we will regard a cyclic word $c(w)$ as a pair $c(w) = (k, c(w_s))$. The first entry is an integer counting the number of copies of Δ_n^2 in $c(w)$. The second entry is the rest of the word written in the σ_i. We shall now present an algorithm for the conjugacy problem in terms of $\mathcal{W}_n$ and $\mathcal{G}_n$. We prove, in a set of lemmas, that this algorithm solves the conjugacy problem in B_n.

ALGORITHM 5.1. *Input: A cyclic braid word $c(w)$. Output: A set of cyclic braid words together with an integer that collectively are a unique representative of the conjugacy class of w.*

(1) Apply rule 1 of $\mathcal{W}_n$ as many times as possible.
(2) Test if w is *splittable*, i.e. if it is in the form $w = w_1w_2$ where w_1 commutes with w_2. If it is, separate w_1 and w_2 and treat them separately from now

on. If not, do nothing. Note that we are testing the linear word w and not $c(w)$.

(3) Apply any rule in $\mathcal{G}_n$ or rules 2 to 4 of $\mathcal{W}_n$, in that order of priority, exactly once to each of the separated cyclic braid words, if possible.
(4) Go back to step 2 of the algorithm and continue until there is no braid word which may be split further and no braid word to which any of the rules in $\mathcal{W}_n$ and $\mathcal{G}_n$ are applicable.
(5) The integer k and the set of split braid words are now collectively the unique representative required.

We note that because of the restrictions on the variable S_{n-1}, rule 5 of $\mathcal{W}_n$ and rule 1 of $\mathcal{G}_n$ are identical. It is obvious from the algorithm that if it cyclicly terminates and $\mathcal{W}_n \cup \mathcal{G}_n$ is cyclicly confluent, then the conjugacy problem in B_n is solved by it. Note that the splitting is valid as we have made the replacement of every inverse generator by inverses of elements of the center and generators in step 1, this replacement is the content of rule 1 of $\mathcal{W}_n$.

LEMMA 5.2. *Algorithm 5.1 cyclicly terminates in $O(L(w)^4 n(w)^9)$ time, where w is the initial braid word.*

Proof. As previously argued (see the proof of theorem 3.3), the execution of step 1 of the algorithm terminates after $O(L(w)n(w)^2)$ time and increases the word length to $L_2(w) = L(w) + mn(w)(n(w) - 1) - m = O(L(w)n(w)^2)$.

Testing splittability is a word problem and can thus be done in $O(L_2(w)^2 n(w))$ [**8**]. As no rule, applied after step 1 in the algorithm, increases the length of the word, the number of times that splitting may be done is bounded by $L_2(w)$.

The application of the local rules terminates as previously shown with complexity $O(L_2(w)^2)$ on linear words. They also terminate here because whenever an infinite rewriting (due to the commutation relation) would be possible, we split the braid word into two. Thus the complexity bound applies here as well.

The global rules either decrease or increase the generator index total of the word. Therefore their application must terminate bounded by $O(L_2(w)n(w))$ as total possible generator index count is bounded by this. The impossibility of interference (first lowering and then raising the total) is clear from the fact that all rule overlaps are rules themselves.

As any one application of either a split, local rule or global rule may trigger the application of another one of those three, the complexity of the algorithm is the product of the partial complexities, i.e. $O(L_2(w)^4 n(w)) = O(L(w)^4 n(w)^9)$. □

LEMMA 5.3. *Algorithm 5.1 is cyclicly confluent.*

Proof. By theorem 3.1, $\mathcal{W}_n$ is confluent and thus contains no critical pairs. Algorithm 5.1 uses $\mathcal{G}_n$ as well as $\mathcal{W}_n$. By construction, $\mathcal{G}_n$ resolves all the cyclic critical pairs of $\mathcal{W}_n$ but, as may be easily verified, introduces no further critical pairs or cyclic critical pairs. By lemma 4.7 this is a necessary and sufficient condition for cyclic local confluence. By lemma 5.2, the algorithm cyclicly terminates and so, by lemma 4.6, the algorithm is confluent. □

LEMMA 5.4. *Algorithm 5.1 solves the conjugacy problem in B_n with computational complexity $O(L(w)^4 n(w)^9)$.*

Proof. The algorithm cyclicly terminates with complexity $O(L(w)^4 n(w)^9)$ and the application of the rules is cyclicly confluent and thus satisfies all the criteria. We must now show that the reflexive-transitive-symmetric closure of the rules is equivalent to the braid group with conjugacy. This is obvious apart from the splitting step in the algorithm. That splitting is correct is obvious too but what remains to be proven is if splitting is confluent with respect to all the other rules, i.e. that it does not matter whether we split first and then apply some rules or vice-versa.

Clearly splitting can be modelled using rewrite rules and splitting terminates. Thus it is confluent with respect to the others if it is locally confluent; or, has no critical pairs. The absence of critical pairs can be verified easily but laboriously by checking the local and global rewrite rules. □

The result of the rewrite chain is a normal form for the braid word but not in the usual sense as we have a set of words that make up the normal form. To test equality between two normal forms, we must compare pair-wise the elements of the normal form which can be done in quadratic time in the length of the word and constant time in the braid group index. Moreover, these methods can be applied to many groups other than the braid groups.

References

[1] Alexander, J. W. 1923 *A lemma on systems of knotted curves* Proc. Nat. Acad. Sci. USA **9**, 93 - 95.

[2] Anshel, M., Goldfeld, D., Anshel, I. 1999 *An Algebraic Method for Public-Key Cryptography* Math. Res. Lett. **6**, 1 - 5.

[3] Artin, E. 1925 *Theorie der Zöpfe* (in German) Abh. Math. Sem. Univ. Hamburg **4**, 47 - 72.

[4] Artin, E. 1947 *Theory of braids* Ann. Math. **48**, 101 - 126.

[5] Baader, F. and Nipkow, T. 1998 *Term Rewriting and All That* (Cambridge University Press, Cambridge).

[6] Bangert, P. D., Berger, M. A. and Prandi, R. 2002 *In Search of Minimal Random Braid Configurations* J. Phys. A: Math. Gen. **35**, 43 - 59.

[7] Birman, J. S. 1974 *Braids, Links and Mapping Class Groups* Ann. of Math. Studies 82 (Princeton Univ. Press, Princeton).

[8] Birman, J. S., Ko, K. H. and Lee, S. J. 1998 *A New Approach to the Word and Conjugacy Problems in the Braid Groups* Ad. Math. **139**, 322 - 353.

[9] Cheon, J.H. and Jun, B. (2003): A Polynomial Time Algorithm for the Braid Diffie-Hellman Conjugacy Problem. in Goos, G., Hartmanis, J. and van Leeuwen, J. (2003): Advances in Cryptology - CRYPTO 2003. LNCS 2729. 212 - 225.

[10] Chow, W. L. 1948 *On the algebraic braid group*, An. Math. **49**, 654 - 658.

[11] Coxeter, H. S. M. and Moser, W. O. J. 1957 *Generators and Relations for Discrete Groups* (Springer, Berlin).

[12] Elrifai, E.A. and Morton, H.R. (1994): Algorithms for positive braids. Quart. J. Math. Oxford **45**, 479 - 497.

[13] Franco, N. and Gonzalez-Meneses, J. (2003): Conjugacy problem for braid groups and Garside groups. J. Alg. **266** 112 - 132.

[14] Garside, F. A. 1969 *The braid group and other groups* Quart. J. Math. Oxford **20**, 235 - 254.

[15] Garber, D., Kaplan, S., Teicher, M., Tsaban, B. and Vishne, U. (2004): Length-based Conjugacy Search in the Braid Group. preprint: math.GR/0209267.

[16] Gonzalez-Meneses, J. (2002): Improving an algorithm to solve Multiple Simultaneous Conjugacy Problems in braid groups. preprint: math.GT/0212150.

[17] Huet, G. 1980 *Confluent reductions: Abstract properties and applications to term rewriting systems* J. Assoc. Comput. Mach. **27**, 797 - 821.

[18] Jacquemard, A. 1990 *About the effective classification of conjugacy classes of braids* J. Pure Appl. Al. **63**, 161 - 169.

[19] Johnson, D. L. 1980 *Topics in the Theory of Group Presentations* Lon. Math. Soc. Lec. Notes Vol. 42 (Cambridge Uni. Press, Cambridge).

[20] Lee, S.J. and Lee, E. (2002): Potential Weaknesses of the Commutator Key Agreement Protocol Based on Braid Groups. in Knudsen, L.R. (2002): EUROCRYPT 2002. LNCS 2332. 14 - 28.

[21] Lee, E. and Park, J.H. (2003): Cryptanalysis of the Public-Key Encryption Based on Braid Groups. in Biham, E. (2003): EUROCRYPT 2003. LNCS 2656. 477 - 490.

[22] Miller, C. F. III 1992 *Decision Problems for Groups - Survey and Reflections* in *Algorithms and Classification in Combinatorial Group Theory* ed. by Baumslag, G. and Miller, C. F. III, Math. Sci. Re. Inst. Pub. Volume 23 (Springer, New York), 1 - 59.

[23] Newman, M. H. A. 1942 *On theories with a combinatorial definition of 'equivalence'* Ann. Math. **43**, 223 - 243.