The Word and Conjugacy Problems in B_n

Patrick D. Bangert*

May 23, 2004

Abstract

We apply the theory of rewriting systems to the braid groups to solve the word problem. The theory is extended to cover the rewriting of cyclicly permuted words which can then be applied to the braid groups to solve the conjugacy problem. Finally, this solution to the braid conjugacy problem is shown to be executable in polynomial-time and thus resolves a long-standing question in low-dimensional topology. This algorithm has potential in many applications such as cryptography.

Contents

1 Introduction **2**

2 The Word Problem in B_n **4**

3 Rewriting Systems for Conjugacy Problems **7**
3.1 The Idea for Free Groups 8
3.2 The Cyclic Newman's Lemma 10
3.3 The Cyclic Critical Pair Lemma 11

4 The Conjugacy Problem in B_n **13**

5 Acknowledgements **15**

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

1 Introduction

The braid group can be defined by its Artin presentation,

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

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.

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* ask 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 $a \rightarrow a\sigma_n^{\pm 1}$ 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.

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. 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 [12]. The best solutions known to date are presented in [7, 14].

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]. The practicability of the scheme depends upon the ease of the word problem solution and the secrecy upon the difficulty of the conjugacy problem solution. In other words, the two users must solve a word problem to decrypt the messages and a possible intruder must solve a conjugacy problem.

Due to these applications, the computational complexity of a solution to the word and conjugacy problems is important. Let us denote the length of the braid word a by $L(a)$ and the number of strings (the braid group it belongs to) by $n(a)$. The most efficient word problem algorithm runs in $O(L(a)^2 n(a))$ time

[7] whereas *all* known conjugacy algorithms run in exponential time in both $L(a)$ and $n(a)$. This seems to indicate that the conjugacy and Markov problems are intractable and that the cryptographic algorithm alluded to above works. All attempts at proving the conjugacy problem to be difficult (such as NP-complete for instance) have not met with success. Indeed, as we will show in this paper, a polynomial-time algorithm with many ramifications for this problem exists. This algorithm will thus allow an intruder to decrypt messages in polynomial-time thus breaking the cipher. 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.

We use the method of rewriting systems to establish our algorithms [5, 11, 15, 18]. Briefly, such systems consist of a set of rules (for example $ab \rightarrow ba$) that may be used to "rewrite" a string of symbols into another string (with the previous rule, the string *abab* is rewritten to *baba* and then again to *bbaa*). The system is *locally confluent* if, in the situation where more than one rule is simultaneously applicable, it does not matter which one we choose and it is *confluent* if this ambiguity is immaterial for any number of rule applications. The system *terminates* if the rewriting process of any word finishes in a finite number of steps; that is, a word is eventually reached to which no rule can be applied (above, the state *bbaa* was reached and the rule may no longer be applied to it; it is the terminating state). If it is both confluent and terminating, it is called *complete* in which case a unique normal form (the final terminating form) exists [5]. If the arrows in a complete system are replaced by equal signs (we take the reflexive-transitive-symmetric closure of the rules in question), then the rewriting system (with its normal form) solves the problem of two expressions being equivalent under the said equational system, i.e. it solves the word problem for finitely-presented groups that have a complete rewriting system. A non-complete system may be made complete using Knuth-Bendix completion [19].

It is, in general, undecidable whether a rewriting system terminates or is confluent. If we have a total order (in fact, it can be done with a less restrictive assumption but we do not need that here) on the alphabet of symbols (or generators of our group) we are dealing with and each rewrite rule simplifies the expression with respect to that order, then the system terminates [10]. If the system terminates, it is confluent if and only if it is locally confluent (Newman's Lemma or the Diamond Lemma) [21]. Moreover there is a mechanical method for checking if a terminating system is locally confluent [13].

Given a rewrite system as a set of pairs (left and right-hand sides of rewrite rules) $R = \{l_i, r_i\}$ we define an *overlap* to be a word $w = abc$ such that ab is a particular instance of an l_i (the l_i and r_i may contain variables and so have different instances) and bc is a particular instance of a different l_j for some words a, b and c. This overlap may be rewritten in two distinct ways (using the rules $l_i \rightarrow r_i$ and $l_j \rightarrow r_j$) and the pair is called *critical* if these two ways are not equivalent in the equational system obtained by taking the reflexive-transitive-symmetric closure. It is obvious that a system with critical pairs is not confluent

as the two reducts are not equivalent and it is the statement of the Critical Pair Lemma that a system is locally confluent if and only there exist no critical pairs [17]. If the system has critical pairs, we may add them as an extra rule to the system in an effort to make it complete; this is the essence of Knuth-Bendix completion [19].

2 The Word Problem in B_n

First, we wish to solve the word problem by the outlined methods. We begin with the braid group and form the associated monoid from it by adding the inverse relations to the braid relations and also augmenting this monoid by adding the relation that gives the generator of the center of B_n ($\Delta_n^2 = (\sigma_1\sigma_2\cdots\sigma_{n-1})^n$ [8]),

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

The reason for doing this is that 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 $M^+(B_n)$, then 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 start the machinery described above 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 from the braid, in the process of rewriting, any sub-braid which lies in the center of the braid group B_n. 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 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. By Knuth-Bendix completion and the necessary

manual labor, 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.

Theorem 2.1 *$\mathcal{W}_n$ is complete and thus solves the word problem for B_n.*

Proof. *(termination)* Every application of $\mathcal{W}_n$ simplifies the 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. □

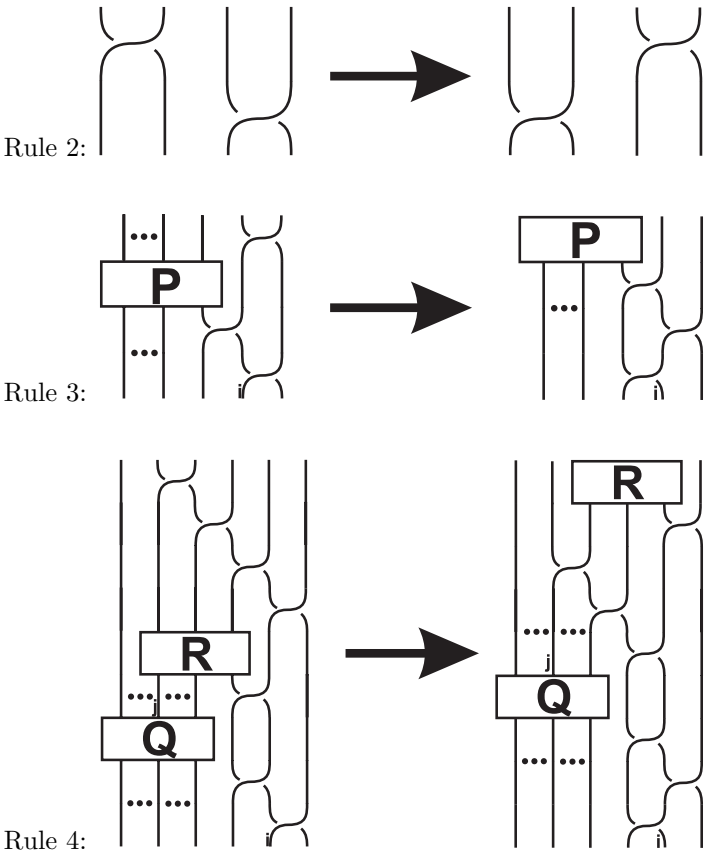

Figure 1: Rules 2, 3 and 4 of TRS $\mathcal{W}_n$ illustrated.

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]. Since $\mathcal{W}_n$ is complete and all strategies are equivalent, we will choose the following strategy.

Algorithm 2.2 *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.

It is clear that algorithm 2.2 solves the word problem from the completeness of $\mathcal{W}_n$ and the fact that once rule 1 is applied as many times as possible, it can not be applied again no matter what other rewrite steps follow as there will be no more inverse generators. From this algorithm, we are able to deduce the computational complexity of this word problem solution.

Theorem 2.3 *$\mathcal{W}_n$ 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 $n(w)^3$ [7]. 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.

3 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.

3.1 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 [16]. 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

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

Thus $\mathcal{R}_w\left(\mathcal{F}_n\right)$ 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 3.1 *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 3.2 *Two words are cyclicly permutable therefore, if one can be moved into the other by cyclic permutation and by word equivalence moves. 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 3.3 *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 [9]. Moreover, if the conjugacy problem is solvable in one representation, it is solvable in all [20]. 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

$$\begin{aligned} w' &\approx f_{s_j}^{p'_j} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} \gamma && (15)\\ &\approx \gamma^{-1}\gamma f_{s_j}^{p'_j} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} \gamma && (16)\\ &\approx \gamma^{-1} w \gamma && (17) \end{aligned}$$

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 idea that gave rise to this definition and the following results is that the troublesome aspect about generalizing a word problem rewrite system to a conjugacy one is the fact that words have a first and last letter where conjugacy allows these to be changed freely. Thus a form is needed that captures this freedom or ambiguity of a beginning of the word and one form that works is that of cyclic permutability. 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).

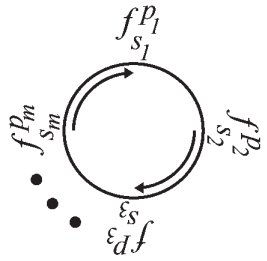

Figure 2: The word w given in equation (8) bent into a circle. While the circularity removes the notions of beginning and end of a word, it preserves the directionality of it.

Definition 3.4 *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 3.5 *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 (including now a further cyclic critical pair). The following section will be devoted to proving these results.

3.2 The Cyclic Newman's Lemma

Newman's Lemma generalizes to the cyclic case without much modification.

Lemma 3.6 *A cyclicly terminating rewrite system is cyclicly confluent if and only if it is cyclicly locally confluent.*

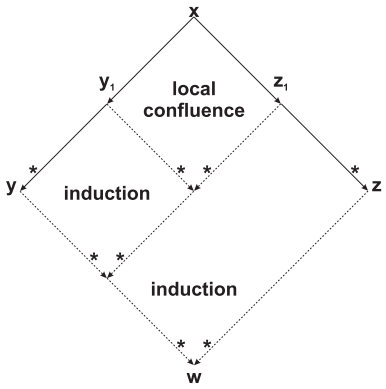

Figure 3: The proof of Newman's Lemma (lemma 3.6) in diagrammatic form. We begin at the top with a local divergence which is rectifiable by assumption and thus by induction any global divergence is also rectifiable. It is because of this diagrammatic proof that Newman's Lemma is also known as the Diamond Lemma.

Proof. This proof is similar to the one given for the standard Newman Lemma in [13]. 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. □

If we can show cyclic termination, we need to check cyclic local confluence. The Critical Pair Lemma can help here.

3.3 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 $\mathcal{R} = \{abxba \rightarrow cxc\}$ 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 TRS's 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 3.7 *A TRS $\mathcal{R} = \{(l_i, r_i)\}$ 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 $l_j \rightarrow r_j$ 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).

□

4 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.

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 $<_b$.

$$
\begin{aligned}
\mathcal{G}_n = \{ & (1)\ \prod_{i=1}^{n-1} (d_{i,1}a_{1,i}S_i) \rightarrow \prod_{i=1}^{n-1} S_i; \\
& (2)\ \sigma_i\sigma_{i-1}P\sigma_i\sigma_{i-1}P' \rightarrow \sigma_{i-1}\sigma_i\sigma_{i-1}P\sigma_{i-1}P' \text{ if } P <_b P' \\
& \text{ or } \sigma_{i-1}\sigma_i\sigma_{i-1}P'\sigma_{i-1}P \text{ if } P' \leq_b P; \\
& (3)\ \sigma_i\sigma_{i-1}Q\sigma_{i-1}R\sigma_i\sigma_{i-1}P \rightarrow \sigma_{i-1}Q\sigma_{i-1}R\sigma_{i-1}\sigma_i\sigma_{i-1}P; \\
& (4)\ \sigma_i\sigma_{i-1}Q_1\sigma_{i-1}R_1\sigma_i\sigma_{i-1}Q_2\sigma_{i-1}R_2 \rightarrow \\
& \sigma_{i-1}\sigma_i\sigma_{i-1}Q_1d_{i-1,j}\sigma_iR_1^+Q_2'\sigma_{i-1}R_2 \text{ if } R_1 <_s R_2 \\
& \text{ or } \sigma_{i-1}\sigma_i\sigma_{i-1}Q_2d_{i-1,j}\sigma_iR_2^+Q_1'\sigma_{i-1}R_1 \text{ if } R_2 \leq_s R_1 \ \}
\end{aligned}
\tag{20}
$$

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 4.1 *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 4.2 *Algorithm 4.1 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 2.3), 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 $O(L_2(w)^2 n(w))$ [7]. As no rule, applied later 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 independently 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 4.3 *Algorithm 4.1 is cyclicly confluent.*

Proof. By theorem 2.1, $\mathcal{W}_n$ is confluent and thus contains no critical pairs. Algorithm 4.1 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 3.7 this is a necessary and sufficient condition for cyclic local confluence. By lemma 4.2, the algorithm cyclicly terminates and so, by lemma 3.6, the algorithm is confluent. □

Lemma 4.4 *Algorithm 4.1 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.

5 Acknowledgements

My warm thanks go to Mitchell Berger, Michael Anshel, Reza Tavakol and Wyn Evans who spent much effort reading and checking the manuscript.

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., 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.

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

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

[10] Dershowitz, N. 1979 *A note on simplification orderings* Inform. Proc. Let. **9**, 212 - 215.

[11] Dershowitz, N. and Jouannaud, J.-P. 1990 *Rewrite Systems* in *Handbook of Theoretical Computer Science Volume B: Formal Methods and Semantics* ed. by van Leeuwen, J. (North-Holland, Amsterdam), 243 - 320.

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

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

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

[15] Jantzen, M. 1997 *Basics of Term Rewriting*, in *Handbook of Formal Languages Volume 3: Beyond Words* ed. by Rozenberg, G. and Salomaa, A. (Springer, Heidelberg), 269 - 338.

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

[17] Jouannoud, J.-P. and Kirchner, H. 1986 *Completion of a set of rules modulo a set of equations* SIAM. J. Comp. **15**, 1155 - 1194.

[18] Klop, J. W. 1987 *Term rewriting systems: A tutorial*, Bulletin of the European Association for Theoretical Computer Science, **32**, 143 - 183.

[19] Knuth, D. E. and Bendix, P. B. 1970 *Simple word problems in universal algebras* in *Computational Problems in Abstract Algebra* ed. by Leech, J. (Pergamon Press, Oxford), 263 - 297. Reprinted in 1983 in *Automation of Reasoning 2* (Springer, Berlin), 342 - 376.

[20] 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.

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

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	$\prod_{i=1}^{n-1} d_{i,1}a_{1,i}S_i$ $S_j = \sigma_{j-1}Q\sigma_{j-1}Rd_{j,k}S'_j$	$\prod_{i=1}^{n-1} S_i$ $S_j = \sigma_{j-1}Q\sigma_{j-1}Rd_{j,k}S'_j$

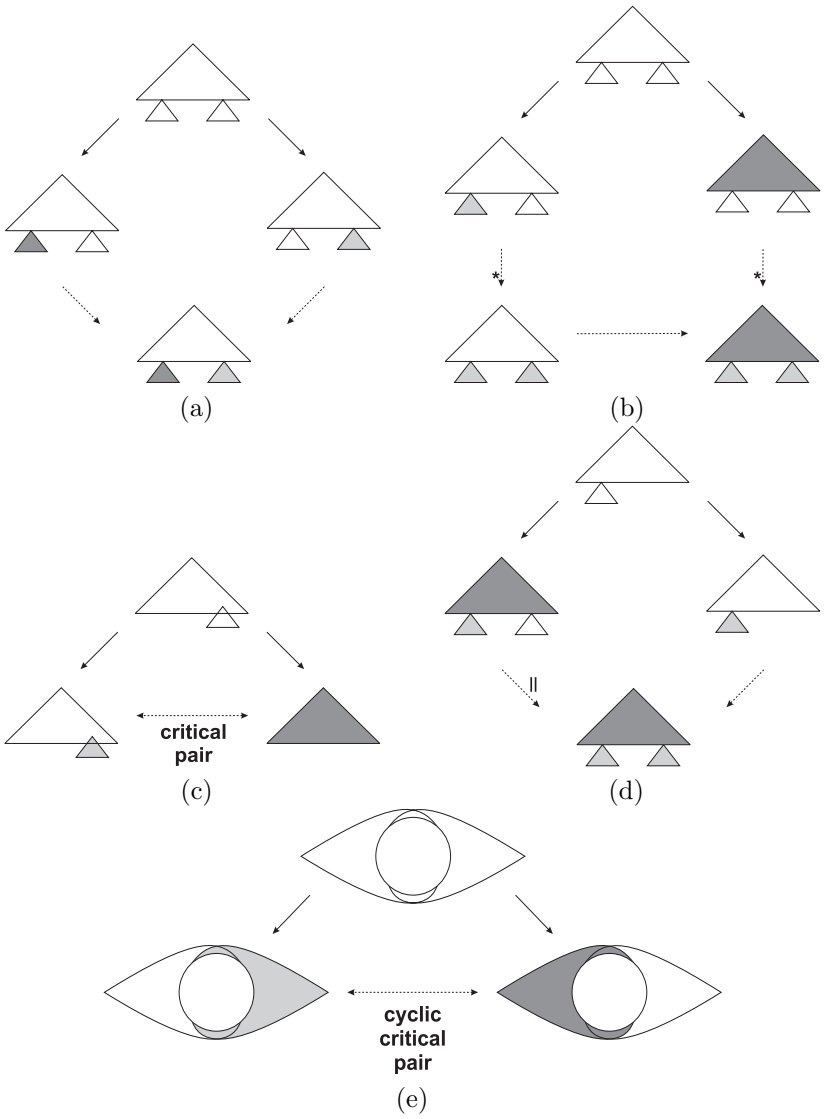

Figure 4: The proof of lemma 3.7 in its four cases: (a) the disjoint case, (b) the variable overlap case, (c) the critical overlap case, (d) the orthogonal case, (e) the circular critical overlap case.

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	$\sigma_i\sigma_{i-1}Q_1\sigma_{i-1}R_1\sigma_i\sigma_{i-1}Q_2\sigma_{i-1}R_2$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Q_1d_{i-1,j}\sigma_i R_1^+Q'_2\sigma_{i-1}R_2$
	& $Q_1 = d_{i-2,j}Q'_1$; $Q_2 = d_{i-2,j}Q'_2$	$\sigma_{i-1}\sigma_i\sigma_{i-1}Q_2d_{i-1,j}\sigma_i R_2^+Q'_1\sigma_{i-1}R_1$