Braids and Knots

Patrick D. Bangert

algorithmica technologies GmbH
Ausser der Schleifmühle 67, 28203, Germany.
Email: `p.bangert@algorithmica-technologies.com`
Internet: `http://www.algorithmica-technologies.com`

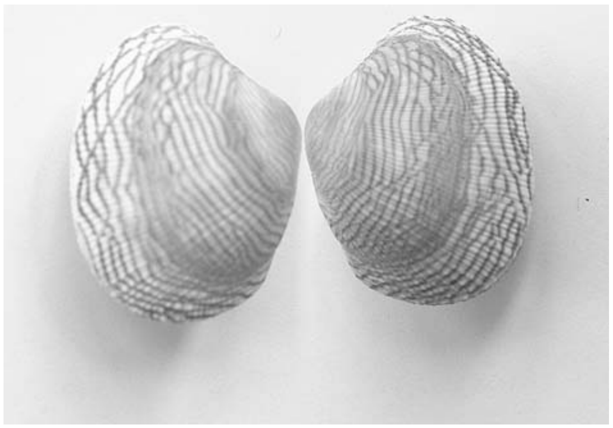

Fig. 1. Shells which display a clearly braided pattern (found by the author in Cetraro, Italy).

Summary. We introduce braids via their historical roots and uses, make connections with knot theory and present the mathematical theory of braids through the braid group. Several basic mathematical properties of braids are explored and equivalence problems under several conditions defined and partly solved. The connection with knots is spelled out in detail and translation methods are presented. Finally a number of applications of braid theory are given. The presentation is pedagogical and principally aimed at interested readers from different fields of mathematics and natural science. The discussions are as self-contained as can be expected within the space limits and require very little previous mathematical knowledge. Literature references are given throughout to the original papers and to overview sources where more can be learned.

A short discussion of the topics presented follows. First, we give a historical overview of the origins of braid and knot theory (1). Topology as a whole is introduced (2.1) and we proceed to present braids in connection with knots (2.2), braids as topological objects (2.3), a group structure on braids (2.4) with several presentations (2.5) and two topological invariants arising from the braid group (2.6). Several

properties of braids are then proven (2.7) and some algorithmic problems presented (2.8).

Braids in their connection with knots are discussed by first giving a notation for knots (3.1) and then illustrating how to turn a braid into a knot (3.2) for which an example is given (3.3). The problem of turning a knot into a braid is approached in two ways (3.4 and 3.5) and then the a complete invariant for knots is discussed (3.6) by means of an example.

The classification of knots is at the center of the theory. This problem can be approached via braid theory in several stages. The word problem is solved in two ways, Garside's original (4.1) and a novel method (4.2). Then the conjugacy problem is presented through Garside's original algorithm (4.3) and a new one (4.4). Markov's theorem allows this to be extended to classify knots but an algorithmic solution is still outstanding as will be discussed (4.5). Another important algorithmic problem, that of finding the shortest equal braid is presented at length (4.6).

We close with a list of interesting open problems (5).

Key words: braids, knots, invariants, word problem, conjugacy problem, rewriting systems, fluid dynamics, path integration, quantum field theory, DNA, ideal knots

1 Physical Knots and Braids: A History and Overview

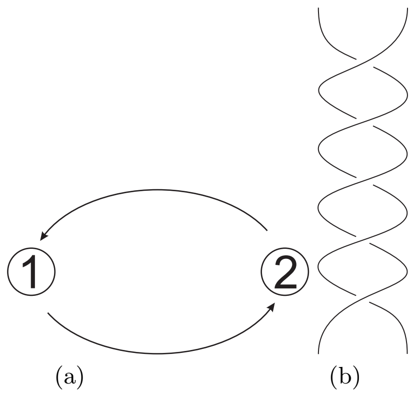

Fig. 2. (a) The simplest method of constructing a braid is to intertwine two strings by exchanging their endpoints and (b) the result of the simple exchange.

Possibly the most important difference between physical and mathematical knots is that mathematics requires the string to be closed. That means that after we tie the knot into a rope, we must glue the ends of the rope together and never undo them. The reason for this is that we are about to consider knots identical if we can continuously deform them into each other. If we had a rope with ends, we could untie the knot and thus every knot would be equal to a segment of straight rope.

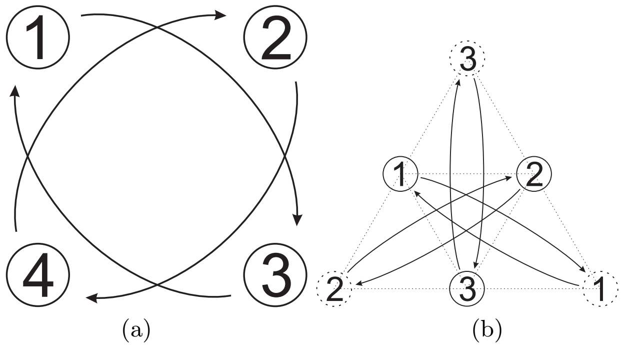

Fig. 3. (a) The points 1 and 3 and the points 2 and 4 form pairs which interchange their positions in turn, thus generating a braid and (b) the three points exchange positions with their image points (drawn in dashed circles) in turn.

The question of how braids are made is interesting in its own right. Suppose that we have n strings which are fixed at one end (we shall call this the *top* end) on a straight line and hang down vertically. The other ends are free to move in a horizontal plane P (the *bottom* end) below the top end. We further label each string by a number from one to n in order from left to right at the top end. Let the intersection of string i and plane P be labelled i also. We may now discuss the braid construction as a series of moves of the points 1 to n in the plane P relative to each other.

5cm

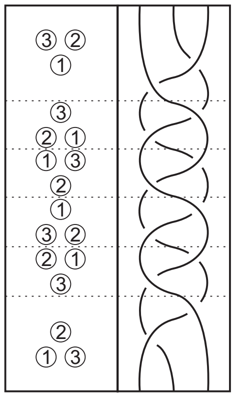

Fig. 4. The braid which results from the motion described in figure 3 (b) with the relative positions of the points at the bottom of each horizontal section.

Let us begin with two points in P which interchange positions at every move, see figure 2 (a). This generates a braid on two strings which looks like figure 2 (b). The natural next step is to consider four points arranged in a square which interchange across the diagonals in turn, see figure 3 (a). This construction method is identical to taking the outer string of a four strand braid and passing it over two and under the string just overcrossed. We alternate between using the left and the right outer string for doing this.

Another way to generalist the scheme of figure 2 (a) is to introduce image points, see figure 3 (b). The image points differ from real points in that there are no strings attached to them. We exchange point and image point in numerical order. The way of constructing a braid shown in figure 3 (b) is one in which each string moves to a point which is its reflected image across the line joining the other two points. This is how the configuration naturally embeds itself into an equilateral triangle.

It is not easy to see what the resultant braid for the construction in figure 3 (b) is so we have drawn it together with the relative positions of the points in figure 4. Note that each exchange of a point with its image point in this scheme generates two crossings in the braid. What is interesting in this example is that we require six exchanges before the points in the plane return to their original relative positions while the braid pattern repeats itself after only two exchanges. The three dimensional structure is thus simpler than the two dimensional dynamical system which gives rise to it. This is an interesting property which can be exploited to classify fundamentally distinct motions in dynamical systems. It should be mentioned that the braid construction of figure 3 (b) is the most optimal way to stir a dye into a liquid [26]. The three points would be rods or paddles of some kind which would be submerged in the liquid and follow the motion prescribed. If one were to record their positions over time then one would obtain figure 4 where the time axis runs vertically upwards.

2 Braids and the Braid Group

2.1 The Topological Idea

Topology is a branch of mathematics that studies the shape of objects independent of their size or position. If we can deform one object into another by a continuous transformation, then we shall call these objects *topologically equivalent*. In everyday terms this means that we may bend, stretch, dent, smoothen, move, blow up or deflate an object but we are not allowed to cut or to glue anything. An example of what we may do is shown in figure 5 in which it is shown by an explicit deformation that one can get from a linked structure to an unlinked structure.

Take for example the doughnut (also called the torus) and the sphere. These two objects are topologically inequivalent. This can be seen easily by observing that the doughnut has a hole while the sphere does not. We can only get rid of the hole by a discontinuous transformation, i.e. in the process of transforming the doughnut into the sphere there will be an instant at which the hole disappears. This is not allowed in topology and so we have motivated that there are topologically different objects.

2.2 The Origin of Braid Theory

Few areas of research can trace their origins as precisely as braid theory. Braid theory, as a mathematical discipline, began in 1925 when Emil Artin published his *Theorie der Zöpfe* [6]. A few problems in this first paper were quickly corrected [7] and the study was made algebraic soon thereafter [25]. As we shall see throughout this chapter, braids are closely related to knots and we need to look at knots to appreciate braids fully.

Knot theory was started in the 1860's by Peter Guthrie Tait, a Scottish mathematician, who endeavored to make a list of topologically distinct knots in response to a request by William Thompson (later Lord Kelvin)

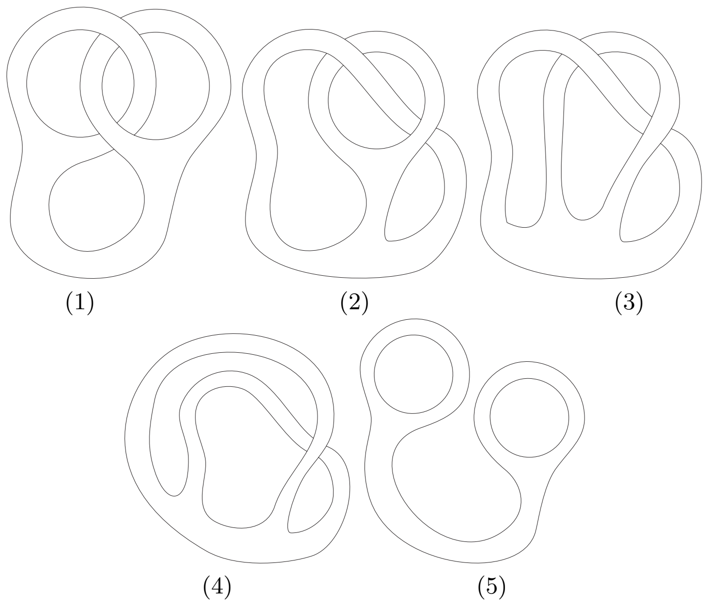

Fig. 5. This sequence of pictures shows how an initially linked structure (1) is slowly transformed into an unlinked structure (5) by a continuous transformation three stages of which are shown.

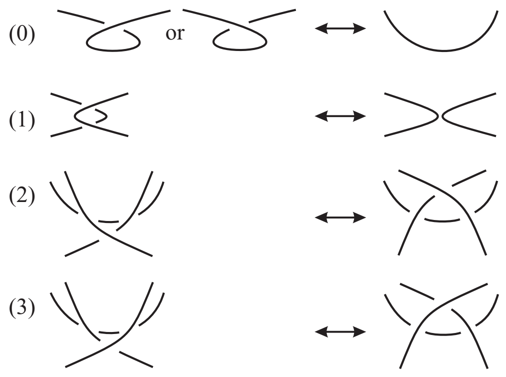

Fig. 6. The Reidemeister moves.

who thought that knotted vortex tubes in the luminiferous ether would make a good model for the elusive atomic theory. This physical application was abandoned when it became clear the the ether did not exist through the Michelson-Morley experiment in 1887. In spite of this, the mathematics was here to stay. Tait first published his work in 1877 at which time he was able to present a long list of knots. It was his purpose to construct the list in order of increasing number of crossings in the knot.

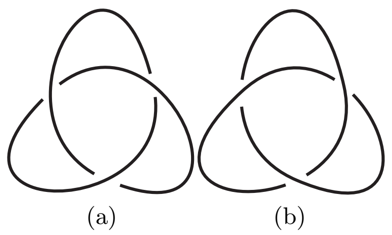

Fig. 7. The (a) trefoil knot and (b) its mirror image.

How can we tell if a knot is the same as another? Reidemeister has provided us with a convenient way to tell. He proved that the moves in figure 6 are sufficient to get from any diagram of a knot to any equivalent diagram [65]. Reidemeister's moves are extremely simple, it is almost obvious that they are sufficient to move between any two equivalent diagrams. Actually producing a sequence of moves for two particular diagrams however, is a non-trivial task. As a result, the moves have found their main use in proofs that certain quantities are identical for all diagrams of a particular knot. We call such quantities *invariants* and they can be integers, real numbers, polynomials, groups, manifolds and other mathematical objects. For example the number of components or closed loops of a knot is an integer invariant as the Reidemeister moves do not perform surgery (cutting or gluing).

Exercise 2.1. Convince yourself that any topological equivalence move possible on a knot can be reduced to Reidemeister moves.

There is only one knot of no crossings – a simple loop of rope – which is called the *unknot* (recall that a mathematical knot exists on rope without ends). No knot of one crossing can exist as this would simply be a twist of one end of the unknot and can just as easily be undone. This twist is the Reidemeister move zero in reverse, see figure 6. The same applies to any knot of two crossings. Matters become more challenging with three crossings because we hit the trefoil knot.

The trefoil knot (see figure 7 (a)) is the knot we usually tie into a shoelace before tying a bow on top of it. Is the trefoil knot the only knot of three crossings? We investigate this question by drawing three double points in the plane (do not differentiate between over and undercrossings yet) which are the intersections of two short line segments each. Then connect the endpoints of the line segments in all possible ways without causing further crossings in the plane. You will find that all of these will unravel to give the unknot by Reidemeister moves of type zero except the trefoil knot. The trefoil knot comes in two natural flavours: the standard type (figure 7 (a)) and the mirror image of the standard type (figure 7 (b)). The *mirror image* of a knot is obtained by switching all of its crossings (see figure 7 (b)). This method is essentially the one which Rev. Kirkman used in the 19^{th} century to construct a list of all knots up to and including ten crossings. The labour involved in this task is prodigious. To complete the list of all knots of three crossings we must ask:

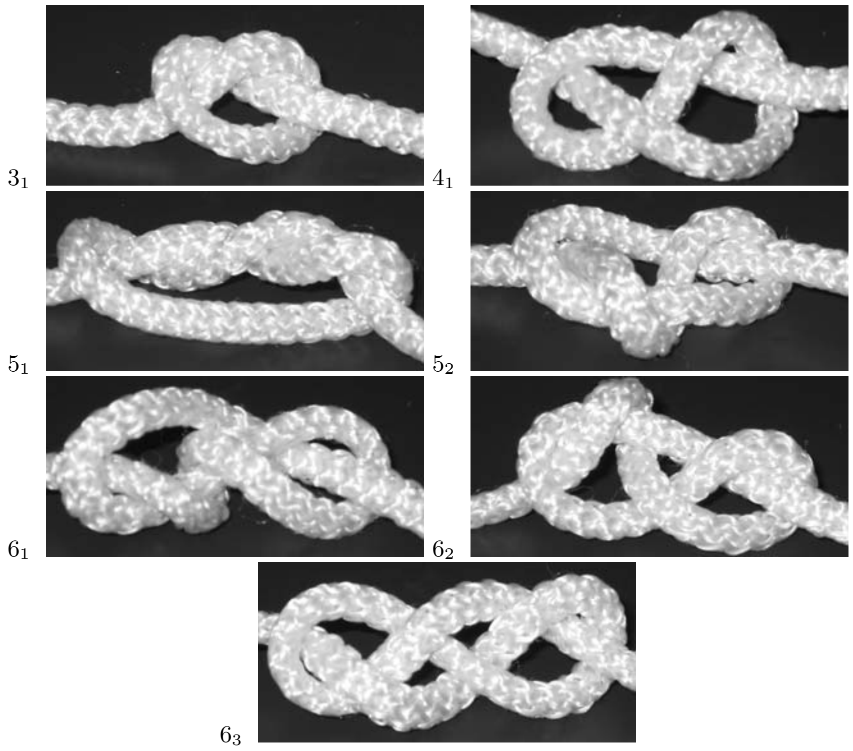

Fig. 8. The prime knots with fewer than seven crossings and their names from standard tables. The knot 3_1 is the trefoil knot of figure 7 (a).

Exercise 2.2. Is the trefoil equivalent to its mirror image? [Hint: This means that you have to find a continuous deformation of the trefoil into its mirror image if they are equivalent or a proof that it is not possible otherwise. The typical method is to look for an invariant if you suspect that they are not equivalent. If you can show that the value of the invariant is different for the two knots, then you have shown that they are different.]

The answer to exercise 2.2 is that the trefoil is *not* equivalent to its mirror image. This is shown by computing an invariant quantity called *Alexander polynomial* for both knots. We will compute the polynomials for both trefoils in section 2.5.

We will motivate the result here by computing a quantity called *writhe* which is an invariant of all the Reidemeister moves except the zeroth one. Imagine you want to hang a painting on the wall and you are putting a screw into the wall to hold the painting up. You twist the screw clockwise to get it into the wall and counterclockwise when you've made a mistake and wish to get it out again. We consider progress positive and mistakes negative so that a crossing in a knot which is achieved by a clockwise rotation of the hands as they follow the orientation of the knot is assigned a weighting of $+1$; the opposite kind is assigned a weight of -1. The writhe w of a knot is the sum of the weights over all the crossings. Let us pick the orientation in which the topmost arch on the trefoils in figure 7 points to the left. Then the standard trefoil has $w = 3$ and the mirror image $w = -3$.

Exercise 2.3. Prove that writhe is invariant for all Reidemeister moves except move zero.

Using such methods, it is possible to construct a large table of knots. In figure 8 we show the first seven knots after the unknot. It is understood that the two ends of the rope must be joined to yield the mathematical knot. We present them in this fashion for ease of understanding and practical experimentation.

The question is: Can we find a general method to determine equality or otherwise for any two knots? The answer is yes, but with qualifications. There exists a method due to Waldhausen, Hakken, Hemion and others but it is so inefficient that it is not possible to use for knots for which we do not know the answer already [39]. There exists a theorem due to Alexander which states that every knot can be represented by a braid [4] which we prove in theorem 2.9. This gave the motivation for people to study braids in order to try to help classify knots. The greatest thrust came from Markov who proved a result for braids similar to Reidemeister move result for knots [55]. Using Markov's theorem to classify knots has proven difficult however and the search continues.

Braids have proven tremendously useful in spite of the fact that they have not lead to a complete knot classification scheme. Many invariant of knots are naturally defined on braids. The most revolutionary invariant, the Jones polynomial, was discovered using braid theory. Beyond this, braids have many applications to various fields as we shall discover in the sections to come.

An operation to combine knots can be defined which we are going to call knot addition and denote it by #.

Definition 2.4. *Given two knots K and L, we define the* knot sum $K\#L$ *as the knot obtained by cutting both K and L at a random location and gluing them together with respect to their orientations.*

This is a simple operation but it is not at all obvious that it is well defined. One can show that: (1) The sum is independent of the points on K and L chosen as cutting points [60], (2) any knot can be uniquely factorized into a finite length sum of knots [67], (3) this sum may actually be determined [68]. Property (1) makes the concept well-defined. The second property establishes the existence of *prime knots*, i.e. knots that may not be decomposed into the sum of others and also that classifying prime knots will classify all knots. The third property means that this is, at least in theory, possible to actually compute. However, the algorithm to find the unique decomposition is the algorithm alluded to previously and thus this is not a practical method. It can be shown that there does not, in general, exist an inverse to the operation of addition of knots. As one may show that knot addition is associative, knots form a semi-group but not a group under the operation of addition [60].

2.3 The Topological Braid

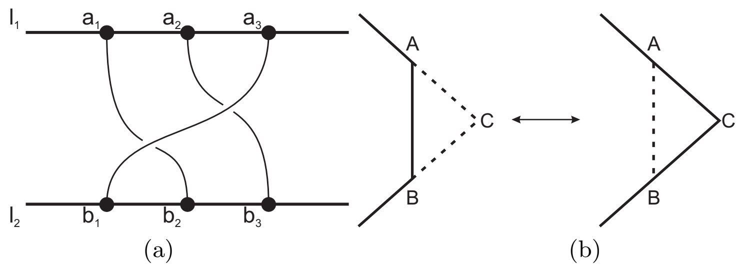

Fig. 9. (a) An example of the definition of a topological braid (see definition 2.5) and (b) an elementary deformation as defined in definition 2.6.

We know from section 1 what a braid is. Mathematically, we have to be slightly more careful.

Definition 2.5 (n-braid). *Let l_1 and l_2 be two parallel lines in a plane P and let $A = \{a_1, a_2, \cdots, a_n\}$ and $B = \{b_1, b_2, \cdots, b_n\}$ be sets of points on l_1 and l_2 respectively. An n-*braid *is a set of (possibly oriented) n non-intersecting polygonal curves which have exactly one endpoint in A and one in B such that all points in A or B are the endpoints of exactly one of these curves and such that any line l parallel to l_1 and l_2 crosses any curve in at most one point.*

An example of a 3-braid, in which we have labelled the lines l_1 and l_2 as well as the point sets A and B, is shown in figure 9 (a); note that the plane P is understood to be the plane of the paper. In this example, we have 3 curves each of which have two endpoints, one in A and one in B. These curves go from l_1 to l_2 monotonically, they do not double back on themselves. This is the meaning that no line parallel to l_1 and l_2 may cross any curve in more than one point. In fact it is this requirement that makes braids substantially simpler than knots and allows a group structure to be defined on braids. An n-braid in the form of definition 2.5 is also called an *open braid*. We shall drop the n from n-braid when no confusion can arise.

The normal braid which is braided into peoples' hair fulfills these requirements. One end of all the strings is fixed on the person's head and others are held in place by some form of rubber band. The braiding in the middle is done in a way that each bundle of hairs goes from top to bottom monotonically. Some Celtic designs used as borders in the Book of Kells or other illuminated books or more commonly used as trimmings for medieval clothing, necklaces, pendants and belts are not usually braids conforming to this definition as their strings often return to a point close to their origin and thus contain a local maximum or minimum.

Whenever we define a new mathematical object, we desire an equivalence relation for the possible instances of this object. As braid theory is a topological pursuit, we will allow ourselves the usual freedom of topology which means that we will allow the object to be distorted in any way as long as this can be done continuously. So we may bend, stretch and pull a string but we may never cut a string or glue two strings together; such actions are called *surgery* and are said to change the topology. For braids, it is clear that if we do not fix the endpoints of the curves, we shall be able to transform any braid into any other yielding a rather boring theory; thus we also require the ends to be fixed. We will call the equivalence relation for braids under these conditions *isotopy*. Before we can define isotopy, we must define what we mean by a topological deformation. An elementary deformation is the basis for all topological deformations.

Definition 2.6 (elementary deformation). *Suppose that a braid string (recall that it was defined as a polygonal curve) has points A and B as vertices. We may then create a further point C, delete the segment AB and create the segments AC and CB. This deformation, and its inverse, is called an* elementary deformation *if and only if the triangle ABC does not intersect any other strings and only meets the current string along its side AB. See figure 9 (b) for an illustration.*

Definition 2.7 (braid isotopy). *Two braids α and β are called* isotopic*, denoted $\alpha \approx \beta$, if and only if α can be transformed into β using a finite number of elementary deformations.*

Suppose we were to label the string which intersects the point b_i by i. On l_2, the string labels from left to right would thus be in numerical order whereas on l_1 they may not be ordered numerically. If we list the numerical labels of the strings which intersect l_1 from left to right, we obtain a permutation on the set of integers $\{1, 2, \cdots, n\}$. A braid thus induces a permutation on the set of the first n integers. For example, the braid in figure 9 induces the permutation $[2, 3, 1]$. The fact that the induced permutation is an equivalence class invariant follows immediately from the requirement that the endpoints be fixed.

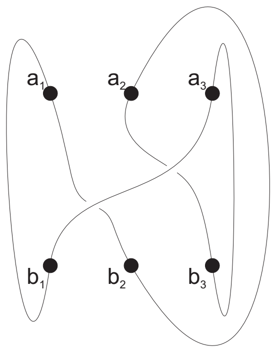

Fig. 10. The braid from figure 9 is closed here. It is immediate upon simple transformation that this knot is the same as the simple loop or the unknot.

An open braid may be *closed* to yield a knot. See figure 10 for the closure of the braid from figure 9. A *closed braid*, denoted $\overline{\alpha}$, is obtained from an open n-braid α by deleting l_1 and l_2 and connecting points a_i and b_i with non-intersecting polygonal curves in P for all $i: \ 1 \leq i \leq n$. It is clear that the closure of any braid yields a knot. Thus some knots may be represented as closed braids. Unfortunately determining the closed braid, given a knot, is not so easy. It is however possible and we shall solve this problem in sections 3.4 and 3.5. The proof of the fact that *all* knots may be represented as closed braids, Alexander's theorem, gave the initial momentum for studying braids in detail [4]. Artin took up the challenge and constructed a theory of braids with a view to use them to deal with knots.

Exercise 2.8. A simple knot invariant is the number of components a knot has. Show that the number of components of the knot $\overline{\alpha}$ is equal to the number of cycles in the permutation that the braid α induces.

Alexander's theorem is usually proved by giving a topological method with which to deform a knot into a closed braid. There exist several distinct methods of doing so but most are not suitable for use; they are only employed to establish the theorem. The proof given here is fundamentally different and new to the best knowledge of the author. The theorem assumes that the knot is oriented but if not the transformation is accurate up to orientation change.

Theorem 2.9 (Alexander [4]). *Every knot may be represented as a closed braid.*

Proof. Consider an oriented straight line a in $\mathbb{R}^3$ which we will call the axis. Choose a point O on a and construct a cylindrical polar coordinate system which has O as its origin. The positive z, or upward vertical, direction is directed parallel to a in the direction of its orientation. The polar angle ϕ increases in the counterclockwise direction, as usual. Using this system, the theorem claims that every knot K can be deformed with respect to a in such a way that the polar angle of a point P going along any component of K strictly increases or more simply: As we travel along the knot we will go around the axis a without ever changing our counterclockwise direction. Suppose we have n straight line segments s_i for $1 \leq i \leq n$ with endpoints R_i and S_i such that the polar angle of S_i, $\phi(S_i)$ is is larger than $\phi(R_i)$. We may form any knot by subdividing these segments into a finite number of straight subsegments, moving the endpoints of the subsegments and performing surgery which identifies R_i and S_i for all i. Here, we will form the knot K by keeping R_i fixed and moving the point S_i creating new points $Q_{i,j}$ indexed by j as necessary. Whenever it becomes necessary to move S_i to a position of lower polar angle than the last $Q_{i,j}$ created, move S_i once around a creating a suitable number of points doing so and then continuing. After the required knot is formed, we perform the surgery of identifying R_i and S_i. By definition of a knot as a polygonal curve such a construction is always possible.

An example of this method applied to forming the trefoil knot is given in figure 11. This proves the theorem. □

2.4 The Braid Group

We note that any braid can be represented by a vertical stack of two types of crossing, see figure 12. When all strings are vertical apart from strings i and $i+1$, we will denote this crossing by σ_i or σ_i^{-1} depending on whether string i overcrosses or undercrosses string $i+1$ respectively. It is thus clear that any braid can be specified by a string of these symbols. We agree to the convention that the left to right direction of the symbols representing a braid shall correspond to the upward direction of the braid; that is, the lowest crossing corresponds to the first symbol. This is a convention and some other authors use the opposite convention. While care is required, no serious consequences arise from this choice. For example, the braid in figure 2 (b) is σ_1^5 (the power means that the symbol σ_1 was repeated five times) and the braid in figure 4 is $\left(\sigma_1\sigma_2^{-1}\sigma_1^{-1}\sigma_2\right)^3$. From now on, we shall denote a braid by these symbols.

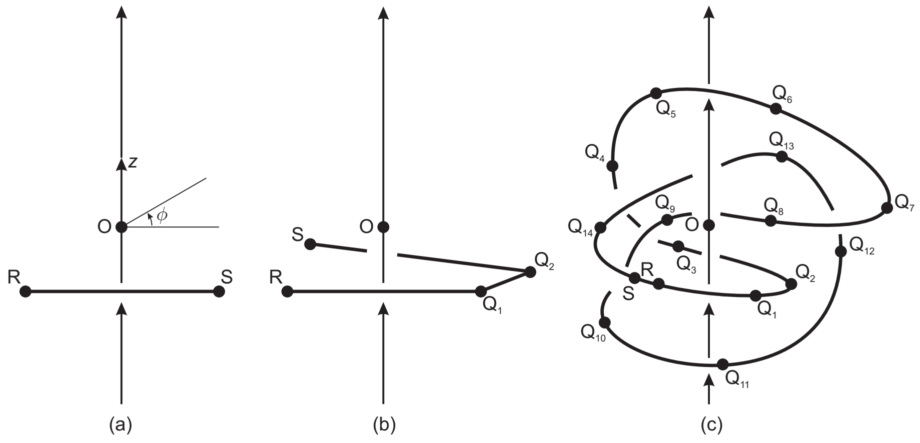

Fig. 11. Constructing the trefoil knot as a closed braid, see proof of theorem 2.9 for a discussion.

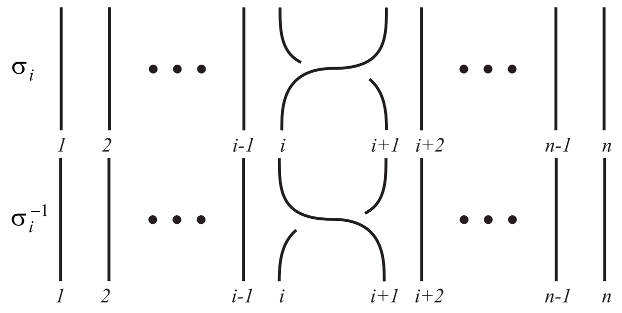

Fig. 12. The generator σ_i and its inverse σ_i^{-1} for the braid group B_n.

Definition 2.10 (braid word). *We will call any sequence of $\sigma_i^{\pm 1}$ a* braid word*.*

Definition 2.11 (positive and negative braid word). *If a braid word contains only σ_i (and no σ_i^{-1}) then it will be called* positive*. However, if it is contains only σ_i^{-1} (and no σ_i) then it will be called* negative*.*

Consider the braid $\sigma_3\sigma_1$ displayed in figure 13 (1a). This braid is clearly topological equivalent to the braid $\sigma_1\sigma_3$ displayed in figure 13 (1b). More generally, every time two neighboring crossings are on distinct pairs of strings, the order in which these crossings are listed in the braid word does not topologically matter. Thus we arrive at the rule that

$$\sigma_i\sigma_j \approx \sigma_j\sigma_i \text{ for } |i-j| > 1 \tag{1}$$

5cm

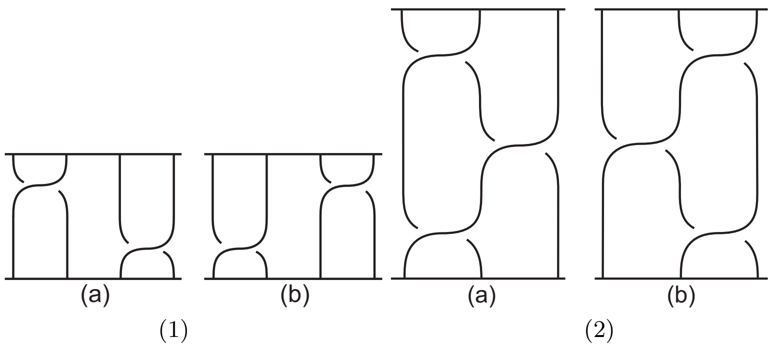

Fig. 13. (1) Two non-interfering crossings can be listed in either order and (2) a crossing may be moved underneath an arch which overcrosses both strings involved

which is usually called the *far commutation* relation as it embodies the fact that generators sufficiently far from each other commute.

It also becomes clear that if we have an arch which over or undercrosses two strings which then cross (see figure 13 (2a)), this crossing may be moved onto the other side of the arch (see figure 13 (2b)). Thus the braids $\sigma_1\sigma_2\sigma_1$ and $\sigma_2\sigma_1\sigma_2$ are topologically equivalent. As this can hold anywhere in a braid, we arrive at the second rule that

$$\sigma_i\sigma_{i+1}\sigma_i \approx \sigma_{i+1}\sigma_i\sigma_{i+1} \tag{2}$$

which is typically called the *braid relation*. We find this name too vague and so we will refer to this relation as the *bridge relation* as it symbolizes that anything not in conflict with the principal pillars may move freely both below and above a bridge.

After some experimentation, one notices that all the moves one may make on a braid while preserving its topology can be reduced to applying the rules in equations 1 and 2 to the braid word. We would like to prove that this is always so.

Theorem 2.12. *(Artin [6] [7]) The equivalence relation upon braid words defined by the relations 1 and 2 is identical to the equivalence relation of braid isotopy (see definition 2.7) upon the braids represented by the braid words.*

Proof. Recall that any knot may be represented as a closed braid. The braid contains all the crossings and Reidemeister's moves define equivalence of knots. Thus braid equivalence is Reidemeister equivalence of the braid. Move zero would create an object which is not a braid but the others apply. Translating these into the σ_i notation and simplifying yields the given presentation.

From figure 12, it is clear that σ_i^{-1} is the inverse of σ_i. As we represent a braid by a braid word in the σ_i from the bottom up, we can easily concatenate two braids together. The braid $\alpha\beta$ is constructed from the braids α and β by identifying the top ends of α with the bottom ends of β. It is obvious that concatenation is associative, i.e. $(\alpha\beta)\gamma \approx \alpha(\beta\gamma)$ and that the concatenation of two braids is a braid. Suppose ι is the braid of n vertical strings and no crossings. We have $\iota\alpha \approx \alpha\iota \approx \alpha$ for any n-braid α. The braid ι acts as an identity. Since we have closure, associativity, inverses and an identity, the set of n-braids forms a group generated by the generators σ_i. We will denote this group by B_n and refer to this family of groups as the *braid groups*. Because of theorem 2.12, B_n has the presentation

$$B_n = \left\langle \{\sigma_1, \sigma_2, \cdots, \sigma_{n-1}\} : \begin{array}{l} \sigma_i\sigma_{i+1}\sigma_i \approx \sigma_{i+1}\sigma_i\sigma_{i+1}, \\ \sigma_i\sigma_j \approx \sigma_j\sigma_i \text{ for } |i-j| > 1 \end{array} \right\rangle \tag{3}$$

It is instructive to consider the braid group from another point of view which curiously leads to the same presentation as given above in equation 3. Consider the space between the two parallel planes which contain l_1 and l_2 respectively after the braid has been removed from it. This space has a fundamental group which may be represented by a series of loops beginning and ending on some randomly chosen base point b and going around the removed braid strings. There are n distinct such loops which we shall call x_i with i running from 1 to n. The fundamental group is free of rank n with the x_i as generators. x_1 is the loop around the first string from the left on each level and so on for the other x_i. From level to level a reassignment of generators becomes necessary. This is called an automorphism and the particular one we need here, a_i is defined by

$$a_i : \ x_i \to x_{i+1}; \ x_{i+1} \to x_{i+1}^{-1} x_i x_{i+1}; \ x_p \to x_p \ (p \neq i, i+1) \tag{4}$$

The map $\alpha : \sigma_i \to a_i$ is a homomorphism of B_n into the automorphism group of F_n, the free group of rank n. It can be shown that the a_i generate a group with presentation identical to equation 3 under the homomorphism α. In other words, a braid word may be regarded as an automorphism of F_n. As the mapping merely consists of a change of symbol for the generators, it is frequently useful not to make a distinction between a braid word as an element of B_n and as an automorphism of F_n. In the next section, we will introduce some other presentations of the braid group.

2.5 Other Presentations of the Braid Group

The presentation of B_n given in equation 3 is called the *Artin presentation* as it was Artin who first used it in the paper which founded the field. The fact that braids admit a group structure simplifies their treatment tremendously. It can be shown that knots do not admit a group structure and this is one reason why the problem of deciding if two knots are equal is so different from the similar question about braids. It is difficult, however to extract useful information from a presentation of a group. For this reason it is useful to search for other presentations of the same group with special properties.

The Artin presentation has the appeal that it is very topological. It is easy to draw the braid given the braid word and it is easy to read off the braid word from a braid. Furthermore, both the far commutation and the bridge relations are simple to perform. One disadvantage is that each braid group has a different number of generators. Consider putting $a = \sigma_1\sigma_2 \cdots \sigma_{n-1}$ and $\sigma = \sigma_1$. After some manipulation, we find the presentation

$$B_n = \left\langle \{a, \sigma\} : a^n \approx (a\sigma)^{n-1},\ \sigma a^{-j}\sigma a^j \approx a^{-j}\sigma a^j \sigma \text{ for } 2 \leq j \leq \frac{n}{2} \right\rangle \tag{5}$$

which we call the *Coxeter presentation.* The Coxeter presentation has the advantage that all braid groups have just two generators but we have lost some of the topological correspondence.

It would be nice to have a matrix representation of the braid groups. To this end, we will represent the identity matrix of n rows and columns by I_n (recall that the identity matrix has unity entries in the leading diagonal and zeros everywhere else). Then we define the mapping $\phi_n(\sigma_i)$ of an Artin generator σ_i to a matrix of n rows and n columns whose entries are Laurent polynomials in the variable t (Laurent polynomials allow both positive and negative powers of the variable). Expressed formally, this means

$$\phi_n : B_n \to GL\left(n, \mathbf{Z}\left[t^{\pm 1}\right]\right) \tag{6}$$

where $\mathbf{Z}$ is the ring of polynomials. The mapping is defined by

$$\phi_n(\sigma_i) = \begin{pmatrix} I_{i-1} & 0 & 0 & 0 \\ 0 & 1-t & t & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & I_{n-i-1} \end{pmatrix} \tag{7}$$

It can easily be shown that ϕ_n is a homomorphism, i.e. that $\phi_n(\sigma_i\sigma_j) = \phi_n(\sigma_i)\phi_n(\sigma_j)$ where multiplication in $GL\left(n, \mathbf{Z}\left[t^{\pm 1}\right]\right)$ is the usual matrix multiplication. A representation is termed *faithful* when the mapping ϕ giving rise to it is injective, i.e. when $x \neq y$ implies $\phi(x) \neq \phi(y)$. This representation is faithful for $n \leq 3$ [54] and not for $n \geq 5$ [13]. For $n = 4$ the answer is unknown.

It should now be easy to write down a matrix representation for any braid word. For example, if $n = 3$, then for $\phi_n(\sigma_1\sigma_2)$ we have

$$\phi_n(\sigma_1)\phi_n(\sigma_2) = \begin{pmatrix} 1-t & t & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1-t & t \\ 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 1-t & t-t^2 & t^2 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \tag{8}$$

This representation is called the *Burau representation.*

Recently a new presentation was invented by Birman, Ko and Lee [15] which they used to solve the word and conjugacy problem in a new way. The generators a_{kl} are defined by

$$a_{kl} = (\sigma_{k-1}\sigma_{k-2}\cdots\sigma_{l+1})\,\sigma_l\,(\sigma_{l+1}^{-1}\sigma_{l+2}^{-1}\cdots\sigma_{k-1}^{-1}) \tag{9}$$

Topologically this is a crossing between two arbitrary braid strings k and l. In a_{kl}, string k overcrosses string l and both strings overcross all other string in between them to be able to cross and then overcross the in between strings again to return to their original (but now switched) positions. Using these generators, B_n has the presentation

$$B_n = \left\langle \{a_{ts}; n \geq t > s \geq 1\} : \begin{array}{l} a_{ts}a_{sr} = a_{tr}a_{ts} = a_{sr}a_{tr}, \\ a_{ts}a_{rq} = a_{rq}a_{ts} \text{ for} \\ (t-r)(t-q)(s-r)(s-q) > 0 \end{array} \right\rangle \tag{10}$$

which we call the *band-generator presentation.* In the Artin representation, we number the strings from left to right on each level. Suppose we were to label each string with a unique label which it would carry throughout the braid. At the bottom of the braid, we number the strings from one to n as we go from left to right. Each crossing in which string i overcrosses string j is labelled with the generator g_{ij}. This presentation is called the *colored braid* presentation as the string labels act like each string was made of a separate color. This representation retains the complete information of the braid but it is not immediately apparent whether a crossing is positive or negative in the Artin sense. Note that this feature means that the generators g_{ij} are self inverse, $g_{ij}^2 \approx e$ the identity. We easily write down the braid group relations in this presentation to get

$$B_n = \left\langle \{g_{ij}\} \text{ for } \begin{array}{l} 1 \leq i, j \leq n \\ i \neq j \end{array} : \begin{array}{l} g_{ij}g_{ik}g_{jk} = g_{jk}g_{ik}g_{ij}, \\ g_{ij}g_{kl} = g_{kl}g_{ij} \text{ for} \\ (i-k)(i-l)(j-k)(j-l) > 0 \end{array} \right\rangle \tag{11}$$

We note that in the colored braid presentation, the fundamental braid Δ_n takes the form

$$\Delta_n = g_{12}g_{13}\cdots g_{1n}g_{23}g_{24}\cdots g_{n-1\,n} \tag{12}$$

$$= \prod_{i=1}^{n-1}\prod_{j=i+1}^{n} g_{ij} \tag{13}$$

2.6 The Alexander and Jones Polynomials

The Burau representation of the braid group is important in the definition of a revolutionary knot-invariant called the *Alexander polynomial.* We begin with a braid word α, construct its Burau representation $\phi_n(\alpha)$ and take the determinant of the matrix $[\phi_n(\alpha) - I_n]_{1,1}$ where the subscript indicates that the first row and column should be deleted. This determinant can be shown to be a topological invariant of the closure of the braid α and is denoted by $\triangle_{\overline{\alpha}}$ such that

$$\triangle_{\overline{\alpha}}(t) = \det[\phi_n(\alpha) - I_n]_{1,1} \tag{14}$$

Thus the Alexander polynomial of the braid $\sigma_1\sigma_2$ of the above example is $\triangle = 1$ which happens to be the same value as the Alexander polynomial for the unknot. While $\overline{\sigma_1\sigma_2}$ is actually isotopic to the unknot, we could not conclude this from its Alexander polynomial. The Alexander polynomial is an *incomplete* invariant in that we can only say that if $\triangle_K \neq \triangle_{K'}$, then $K \neq K'$. The converse is not true, in general. Nevertheless, the Alexander polynomial is very important in knot theory. Let us compute the Alexander polynomial for both trefoils (see figure 7). The ordinary trefoil is the closure of σ_1^3 and its mirror is the closure of σ_1^{-3}. Now we have

$$\phi_2\left(\sigma_1^3\right) = \begin{pmatrix} (1-t)^3 & 0 \\ 0 & t^3 \end{pmatrix} \qquad \phi_2\left(\sigma_1^{-3}\right) = \begin{pmatrix} (1-t)^{-3} & 0 \\ 0 & t^{-3} \end{pmatrix} \tag{15}$$

and thus

$$\triangle_{\overline{\sigma_1^3}} = t^3 - 1 \qquad \triangle_{\overline{\sigma_1^3}} = t^{-3} - 1 \tag{16}$$

And since the Alexander polynomials of the two trefoils are distinct, the knots must be distinct.

Many other polynomials invariants have been devised after the Alexander polynomial, the most important is the Jones polynomial. The Jones polynomial is also an incomplete invariant of knots which was originally constructed in terms of braids. We shall not prove that it is an invariant, nor that it is incomplete but we shall give an easy method to obtain it. Even though it is incomplete, it is a very powerful invariant in that it distinguishes many knots not distinguished by other invariants such as the Alexander polynomial.

We define the *tensor product* of a $p \times q$ matrix A and a $r \times s$ matrix B by

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1q}B \\ a_{21}B & a_{22}B & \cdots & a_{2q}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1}B & a_{p2}B & \cdots & a_{pq}B \end{pmatrix} \tag{17}$$

$A \otimes B$ is clearly a $pr \times qs$ matrix. We also define the *trace* of a $n \times n$ matrix C by

$$tr(C) = \sum_{i=1}^{n} c_{ii} \tag{18}$$

Consider the matrix

$$\mu(t) = \begin{pmatrix} 1 & 0 \\ 0 & t \end{pmatrix} \tag{19}$$

and define

$$\mu^{\otimes n} = \underbrace{\mu \otimes \mu \otimes \cdots \otimes \mu}_{n \text{ times}} \tag{20}$$

then $\mu^{\otimes n}$ is a $2^n \times 2^n$ matrix and $tr\left(\mu^{\otimes n}\right) = (1+t)^n$. Furthermore let

$$R = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -t^{1/2} & 0 \\ 0 & -t^{1/2} & 1-t & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}; \qquad R^{-1} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1-t^{-1} & -t^{-1/2} & 0 \\ 0 & -t^{-1/2} & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{21}$$

where we use the convention that $\left(t^{1/2}\right)^2 = t$.

After all these definitions, we are ready to define a map from the braid group B_n to the $2^n \times 2^n$ matrices with Laurent polynomial entries ($\Phi_n : B_n \to M\left(2^n, \mathbb{Z}\left[t^{1/2}, t^{-1/2}\right]\right)$) given by

$$\Phi_n\left(\sigma_i^\epsilon\right) = \underbrace{I_2 \otimes I_2 \otimes \cdots I_2}_{i-1 \text{ times}} \otimes R^\epsilon \otimes \underbrace{I_2 \otimes I_2 \otimes \cdots I_2}_{n-i-1 \text{ times}} \tag{22}$$

where $\epsilon = \pm 1$ and I_2 is the 2×2 identity matrix. For some general braid word β we have

$$\beta = \sigma_{i_1}^{\epsilon_1} \sigma_{i_2}^{\epsilon_2} \cdots \sigma_{i_k}^{\epsilon_k} \qquad (1 \leq i_1, i_2, \cdots, i_n < n;\ \epsilon_j = \pm 1) \tag{23}$$

$$\Phi_n(\beta) = \Phi_n\left(\sigma_{i_1}^{\epsilon_1}\right) \Phi_n\left(\sigma_{i_2}^{\epsilon_2}\right) \cdots \Phi_n\left(\sigma_{i_k}^{\epsilon_k}\right) \tag{24}$$

If we now define

$$\epsilon(\beta) = \epsilon_1 + \epsilon_2 + \cdots + \epsilon_k \tag{25}$$

then the Jones polynomial is given by

$$V_{\overline{\beta}}(t) = \frac{t^{\frac{\epsilon(\beta)-n+1}{2}} tr\left(\Phi_n(\beta)\mu^{\otimes n}\right)}{1+t} \tag{26}$$

and we have that if $\overline{\beta_1} \approx \overline{\beta_2}$, then $V_{\overline{\beta_1}}(t) = V_{\overline{\beta_2}}(t)$ for any two braids β_1 and β_2 (not necessarily members of the same braid group).

Exercise 2.13. Compute the Jones polynomial for the 2-braid σ_1, the closure of which is clearly the unknot (solution follows).

We just compute and obtain

$$V_{\overline{\sigma_1}}(t) = \frac{t^{\frac{1-2+1}{2}}}{1+t} tr \left[\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -\sqrt{t} & 0 \\ 0 & -\sqrt{t} & 1-t & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & t & 0 & 0 \\ 0 & 0 & t^2 & 0 \\ 0 & 0 & 0 & t^3 \end{pmatrix} \right] = 1 \tag{27}$$

It is a well-known conjecture that the Jones polynomial recognizes the unknot, that is to say that the Jones polynomial is equal to 1 if and only if the knot is isotopic to the unknot. Many people have been searching for a proof and a counterexample and none has been found thus far. Resolving this question is one of the more important open problems in knot theory.

Proving that the Jones polynomials is an invariant of closures of braids is easy. For reasons of space, we simply outline the equations that need to be checked by straightforward calculations.

Theorem 2.14. *The Jones polynomial is a knot invariant.*

Proof. By direct calculation, check

1. The mapping Φ_n is a homomorphism from B_n to $M\left(2^n, \mathbb{Z}\left[t^{1/2}, t^{-1/2}\right]\right)$. This can be done by checking that
$$\Phi_n\left(\sigma_i\sigma_j\right) = \Phi_n\left(\sigma_j\sigma_i\right); \qquad \Phi_n\left(\sigma_i\sigma_{i+1}\sigma_i\right) = \Phi_n\left(\sigma_{i+1}\sigma_i\sigma_{i+1}\right) \qquad (|i-j|>1) \tag{28}$$
2. By Markov's theorem, two closed braids are isotopic if and only if they can be reached from each other by conjugacy and stabilization. Thus we need to check if the Jones polynomial is invariant under these two moves,
$$V_{\overline{\gamma\beta\gamma^{-1}}}(t) = V_{\overline{\beta}}(t); \qquad V_{\overline{\beta}}(t) = V_{\overline{\beta\sigma_n^{\pm 1}}}(t) \qquad (\gamma, \beta \in B_n) \tag{29}$$

The polynomial invariants can be calculated using so-called *skein relations.* These relations are expressions relating the polynomials of knots that are identical except for a single local change. In braid language, the Alexander polynomial of the closed braid $\beta \in B_n$, $A_\beta(z)$ is given by the relation
$$A_{\beta\sigma_i}(z) - A_{\beta\sigma_i^{-1}}(z) = zA_\beta(z) \tag{30}$$
together with the boundary condition that if the closure of β is the unknot, $A(z) = 1$. The Jones polynomial $J_\beta(z)$ has the same boundary condition but takes the skein relation
$$\frac{J_{\beta\sigma_i}(z)}{z} - tJ_{\beta\sigma_i^{-1}}(z) = \left(\sqrt{z} - \frac{1}{\sqrt{t}}\right)J_\beta(z) \tag{31}$$
Using these skein relations, it is possible to calculate the polynomial invariants straight from the diagram by successively eliminating crossings [48].

2.7 Properties of the Braid Group

In this section we will prove various propositions which enumerate the basic properties of braids. The Artin presentation will be used for all these as for most of the rest of the chapter.

Recall that the center of a group is the set of all those elements which commute with all other elements in that group. The centre of the braid groups will become important in many places and so we will construct it.

Definition 2.15 (fundamental word). *The* fundamental braid word $\varDelta_n \in B_n$ *is defined by*

$$\varDelta_n = \sigma_1\sigma_2 \cdots \sigma_{n-1}\sigma_1\sigma_2 \cdots \sigma_{n-2} \cdots \sigma_1\sigma_2\sigma_1 \tag{32}$$

It is a simple matter of applying the braid group relations to find that

$$\varDelta_n^2 = (\sigma_1\sigma_2 \cdots \sigma_{n-1})^n \tag{33}$$

Proposition 2.16. *The center $C(B_n)$ of the braid group B_n is the set*

$$C(B_n) = \left\{\varDelta_n^{2i}\right\} \tag{34}$$

for any (positive, negative or zero) integer i.

Definition 2.17. *The* ascending braid word a_j *is defined by* $a_j = \sigma_1\sigma_2 \cdots \sigma_j$.

Proposition 2.18. *In B_n, we have $\sigma_i a_j \approx a_j \sigma_{i-1}$ for $1 < i \le j < n$.*

Proposition 2.19. *For all i we have $\sigma_i \varDelta_n \approx \varDelta_n \sigma_{n-i}$.*

Definition 2.20. *Two positive n-braid words a and b are called* positively equal *if and only if there exists a sequence of words W_i with $0 \le i \le p$ for some finite p for which $W_0 = a$, $W_p = b$, W_i is obtained from W_{i-1} by a single application of one of the defining relations of B_n, and all W_i are positive.*

In other words, two braids are positively equal if we can transform one into the other without ever having to use an inverse generator.

Proposition 2.21. *Two equal positive braid words are positively equal.*

2.8 Algorithmic Problems in the Braid Groups

We have defined braids, elucidated their connection with knots and found a group structure on braids. This group structure has certain properties of which we enumerated a few important ones in the last section. There are a number of questions which we may readily ask about braids, the most significant of which we shall describe in this section. Clearly, we wish to know both necessary and sufficient conditions for equivalence.

Definition 2.22 (word problem). *Given two braid words $\alpha, \beta \in B_n$, the* word problem *asks whether $\alpha \approx \beta$.*

The question of whether $\alpha \approx \beta$ is identical to the question of whether $\alpha\beta^{-1} \approx e$, the identity in B_n. Thus the word problem reduces to recognizing the identity element.

Recall that two elements a and b of some group G are called *conjugate*, denoted $a \approx_c b$, if and only if there exists a $c \in G$ such that $a \approx cbc^{-1}$. The conjugacy condition becomes $\alpha\gamma \approx \gamma\beta$ for two braid words $\alpha, \beta \in B_n$ to be conjugate with respect to a third braid word $\gamma \in B_n$. We are naturally interested under what conditions such a commutation relation exists.

Definition 2.23 (conjugacy problem). *Given two braid words $\alpha, \beta \in B_n$, the* conjugacy problem *asks whether there exists a third braid word $\gamma \in B_n$ such that $\alpha\gamma \approx \gamma\beta$.*

Suppose $\alpha \approx \beta$, then we also have $\alpha\gamma \approx \gamma\beta$ for $\gamma \approx e$. Thus any equal braid words are also conjugate but two conjugate braid words are not necessarily equal and so the conjugacy problem subsumes the word problem.

The most central question in knot theory is that of classification: Given two knots K_1, K_2 are they topologically equivalent $K_1 \approx K_2$? Markov has found it possible to translate this question into a question about braids. Imagine closing the braid cbc^{-1}. We may move the subbraid c through the closure part so that it emerges on top of the rest of the braid, i.e. becomes $bc^{-1}c \approx b$. In other words, conjugation of a braid preserves closed braid isotopy: Any two conjugate braids represent equivalent knots. Conjugation is a sufficient condition but unfortunately not necessary. The braid $e \in B_1$ is just a single vertical line segment which closes to the unknot. The braid $\sigma_1 \in B_2$ is the braid of single positive crossings which also closes to the unknot. As the braids are in different braid groups, they have a different number of strings and are thus not conjugate to each other. Such braids can be related through a move called stabilization.

Definition 2.24 (stabilization move). *Given a braid $\alpha \in B_n$, the operation $\alpha \leftrightarrow \alpha\sigma_n^{\pm 1}$ is called* stabilizing *the braid and is referred to as the* stabilization move*.*

Note that stabilization changes the number of strings in the braid by one and that we refer to both the addition and removal of a string as stabilizing. Starting with $e \in B_1$, we stabilize once to obtain $\sigma_1 \in B_2$. In this way, we can move between two braids in different braid groups. Markov's theorem states that conjugacy and stabilization are enough for knot equivalence.

Theorem 2.25 (Markov [55], Birman [14]). *Given two braids $\alpha \in B_n$ and $\beta \in B_m$, we have $\overline{\alpha} \approx \overline{\beta}$ if and only if β may be obtained from α by a finite sequence of conjugacy or stabilization moves; we denote this by $\alpha \approx_M \beta$ and call α and β* Markov equivalent*.*

In [55], Markov stated this theorem but the first complete published proof appeared in [14]. We shall not prove this theorem here as it is difficult and lengthy and would distract from the flow of the chapter. It is clear that we are interested in approaching knot classification from a braid theory point of view.

Definition 2.26 (Markov problem). *Given two braid words $\alpha \in B_n$ and $\beta \in B_m$, the* Markov problem *or* algebraic link problem *asks whether $\alpha \approx_M \beta$.*

By Markov's theorem, a solution to the Markov problem would provide a solution to the knot classification problem. As the Markov problem subsumes the conjugacy problem, we are presented with a hierarchy of three combinatorial problems in group theory which would have significant weight if a solution were found. Solutions exist to the word and conjugacy problems and we shall develop new solutions in later sections. A solution to the Markov problem only exists in the sense that there exists a knot classification scheme based on 3-manifold classification. This is an indirect and unusable solution as it is exponential in the amount of computing time required as a function of the crossing number of the knots concerned.

As many braid words represent the same braid, we are naturally lead to ask for a particularly simple representative. A very natural definition of "simple" in the case of a braid word is the least length possible; the shorter the word, the simpler it is. As length has an obvious minimum, we formulate the minimal word problem.

Definition 2.27 (minimal word problem). *Given a braid word $\alpha \in B_n$, the* minimal word problem *asks one to find a braid word $\alpha_m \in B_n$ such that $\alpha_m \approx \alpha$ and $L(\alpha_m) \leq L(\alpha')$ for any braid word $\alpha' \approx \alpha$. The word α_m is the* minimal word *as it is, by definition, the shortest word in its equivalence class.*

The word problem is frequently solved by devising an algorithm which finds a unique representative for the equivalence class of the element under consideration. In many groups, this so called *normal form* is also of minimum length in the equivalence class so that word problem and minimal word problem are solved together. Examples of such groups include free groups, HNN-extensions and free products. For the braid groups, we shall find that the minimum word problem is far more complicated to solve than the word problem. In fact, under some very reasonable assumptions, the minimum word problem in the braid groups can only be solved by what amounts to a global search of the equivalence class. We shall outline these assumptions and the methods by which the global search may be done in later sections.

3 Braids and Knots

3.1 Notation for knots

Dowker-Thistlethwaite Code When he originally invented knot theory, Tait sought to construct a table of distinct prime knots. Two tasks had to be undertaken: (1) An exhaustive list of all possible knots had to be constructed and (2) all duplicates had to be struck from that list. Tait used mainly combinatorial methods to construct an exhaustive list and then tried to eliminate duplicates by trying to deform knots into each other. Tait labelled each crossing by a letter of the alphabet. He named a knot by starting at some random

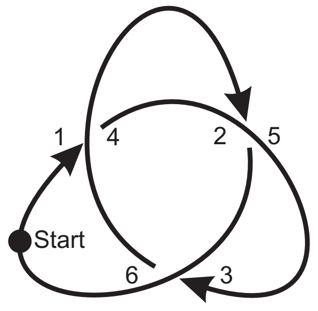

Fig. 14. The procedure for obtaining the Dowker-Thistlethwaite code for the trefoil.

point and going along it in the direction of its orientation, writing down the label of the crossings as he passed them. This string of letters is called the *Tait code* of a knot. Clearly a knot with n crossings has a name consisting of $2n$ letters. There exist a finite, though large, number of possible such names for any n. Not all possible names can arise from naming a knot, however and Tait was able to find methods to determine if a specific name was valid.

Dowker and Thistlethwaite improved Tait's methods and introduced their own naming convention [35]. Consider a knot of n crossings and start at a random point going along the knot in the order of its orientation. Name the crossings in numerical order as you pass them giving each crossing two numerical labels (you will pass each crossing twice before returning to the staring point). Each crossing will have two numbers associated with it, i and j, say. We can construct an involution a from these by putting $a(i) = j$ and $a(j) = i$. Thus we have $a(a(k)) = k$ for any label k. Note that a reverses parity, that is if i is odd and $a(i) = j$, then j is even and vice-versa. We agree to write a_i for $a(i)$, S for the sequence $a_1, a_2, \cdots, a_{2n}$ and S_{odd} for the sequence $a_1, a_3, \cdots, a_{2n-1}$. The sequence S_{odd} completely determines both S and a [35].

There are $2n$ different possible starting points on the knot and if it has no orientation, there are two possible orientations. Not all sequences S_{odd} are identical. The *standard sequence* S_{std} for a knot will be the lexicographical minimum over all possible S_{odd}. Consider, for example, the trefoil knot in figure 14. The starting point is labelled in the figure and we proceed to name the three crossings twice each in the order in which we encounter them. This gives rise to the involution

$$a_1 = 4,\ a_2 = 5,\ a_3 = 6,\ a_4 = 1,\ a_5 = 2,\ a_6 = 3 \tag{35}$$

which results in $S_{std} = 4, 6, 2$.

Dowker and Thistlethwaite were able to determine both necessary and sufficient conditions for a sequence of number to be realizable as a knot. This algorithm is relatively simple and quick to implement and has been used by them to tabulate knots. This code is very compact and easy to obtain from a knot, but their tabulation methods focus on enumerating all possible sequences and so we ask: How may we recover the knot given the code?

We note first of all that a knot of n crossings will get $2n$ labels ($2n$ being necessarily even). Suppose $S_{std} = b_1, b_2, \cdots, b_n$ and the original involution is a. As S_{std} is a standard sequence, we have that $a_{2i-1} = b_i$ for i ranging from one to n. Because of the definition of a, we then have

$$a(a(2i-1)) = a(b_i) = 2i - 1 \tag{36}$$

Since a reverses parity and we list only the a_i for i odd in the standard sequence, all these a_i are even. Thus we have recovered the entire involution a. The crossings of the knot thus get the double names i and b_i for i ranging from one to n. Having gotten the complete labelling information, we can draw the crossings on our paper as double points and connect them in order of the labelling and thus retrieve the knot. We may not get the same number of crossings that we had in the original projection but this does not matter topologically.

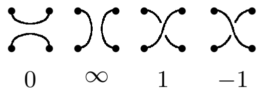

Fig. 15. The four elementary tangles.

Conway's Basic Polyhedra We constructed the Alexander polynomial in section 2.5. Starting from a knot, we must first construct a closed braid isotopic to the knot, then write down its Bureau representation and take its determinant to obtain the Alexander polynomial. In practise this process, particularly constructing a closed braid representative, is very complicated and time-consuming. When Conway looked for a mechanizable method, he was lead to construct a new notation for knots. From this notation, he was able to extract the Alexander polynomial so straightforwardly that he proceeded to calculate them all by hand rather than mechanize the method.

A knot diagram has crossings and arcs connecting the crossings. If we were to draw small circles around the crossings and then ignore what is in the circles, we would have a template for the knot. Into the circles we could insert any of the four elementary tangles of figure 15 to generate several knots, one of which would be the original knot. The numerical names of the elementary tangles arise from their classification which will not concern us here. Conway's notation derived from the observation that many knot diagrams are the same after the crossings have been so removed. In such a way, we may generate a large number of different knots starting from one such knot template and

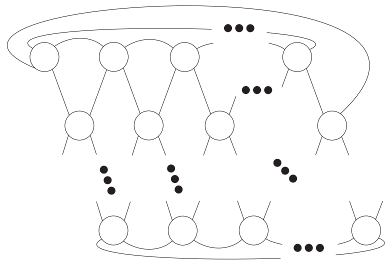

Fig. 16. The universal polyhedron.

inserting different tangles into different slots. Conway called these templates *basic polyhedra* and was able to show that eight basic polyhedra are enough to generate all the different knots up to and including 11 crossings [30].

The number of different basic polyhedra needed to generate the knots of higher crossing number n increases sharply with n and as the polyhedra lack pattern, the next one may not easily be generated from the previous ones. This problem gave rise to the idea of the universal polyhedron to be discussed next.

The Universal Polyhedron The *universal polyhedron* $P(i,j)$ is defined by figure 16. It has i rows and j columns of *vertices* which will be filled with elementary tangles and which are connected by *edges*. It can be shown that any knot can be represented by some $P(i,j)$ and that we may write this down in a matrix form,

$$P(i,j) = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1j} \\ p_{21} & p_{22} & \cdots & p_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i1} & p_{i2} & \cdots & p_{ij} \end{pmatrix} \tag{37}$$

with the elements $p_{kl} \in \{1,-1,0,\infty\}$. For example, the knots up to and including six crossings (see figure 8) are given by

$$\begin{array}{ll} 0_1 = (\infty) & 3_1 = \begin{pmatrix} -1 & -1 \\ -1 & \infty \end{pmatrix} \\ 4_1 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & \infty \end{pmatrix} & 5_1 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \end{pmatrix} \\ 5_2 = \begin{pmatrix} \infty & 0 & 1 \\ -1 & -1 & -1 \\ 0 & 1 & \infty \end{pmatrix} & 6_1 = \begin{pmatrix} \infty & 0 & -1 & -1 \\ 1 & 1 & 0 & \infty \\ 0 & -1 & -1 & \infty \end{pmatrix} \\ 6_2 = \begin{pmatrix} 1 & 1 & 1 & \infty \\ 0 & -1 & -1 & \infty \\ 0 & 1 & \infty & 0 \end{pmatrix} & 6_3 = \begin{pmatrix} 1 & 1 & 1 \\ 0 & -1 & \infty \\ 1 & 1 & 0 \end{pmatrix} \end{array} \tag{38}$$

Theorem 3.1. *Every regular projection of any knot may be represented by the universal polyhedron $P(i,j)$ for some i and j all the vertices of which contain elementary tangles.*

Proof. A regular projection of a knot is characterized by a finite number n of double points and $2n$ arcs which connect the double points in a specific manner. For sufficiently large i and j, the polyhedron $P(i,j)$ can accommodate all double points in the form of ± 1 tangles and can achieve the desired connection of these by placement of 0 and ∞ tangles into it. This is obvious because the 0 and ∞ tangles represent horizontal and vertical connectors in the polyhedron. Because this connection may be achieved without ± 1 tangles, it is clear that no further components, with the possible exception of unknots, are created. Thus what remains to be shown is that no unwanted unknots will be created.

There are ij vertices and $2ij$ edges connecting them in the empty polyhedron $P(i,j)$. Eliminating one vertex by a 0 or ∞ tangle, eliminates two edges. Apart from the ± 1 tangles of which there are n, the final polyhedron will contain $ij-n$ tangles of type 0 and ∞ which will have eliminated $2(ij-n)$ edges from the original polyhedron, leaving exactly $2n$ edges which are needed to connect the double points. Thus there is no extra edge left over which could possibly form an extra component. Therefore any knot may be represented using the basic polyhedron $P(i,j)$ and elementary tangles.

3.2 Braids to Knots

Suppose we have a braid b given by a braid word and we want to denote the knot that is isotopic to its closure in the standard notations. The universal polyhedron (figure 16) turned through π radians becomes a closed braid template if every vertex is filled with tangles of the types 1, -1 and 0.

The n-braid b will be specified by a function $b(t) = \sigma_i^{\pm 1}$ which gives the t^{th} Artin generator of b for $1 \leq t \leq c$ where c is the number of crossings in the braid. The map ξ_i will map an Artin generator $\sigma_j^{\pm 1}$ to the elementary tangles 1 and -1 if the exponent of the Artin generator is 1 and -1 respectively and $i = j$ and will map any Artin generator to the elementary tangle 0 otherwise,

$$\xi_i\left(\sigma_i\right) = 1 \tag{39}$$

$$\xi_i\left(\sigma_i^{-1}\right) = -1 \tag{40}$$

$$\xi_i\left(\sigma_j^{\pm 1}\right) = 0 \text{ for } i \neq j \tag{41}$$

The closed braid $\overline{b}$ can be contained in the polyhedron $P(n-1,c)$ with $p_{ij} = \xi_i\left(b(j)\right)$ with $1 \leq i < n$ and $1 \leq j \leq c$.

For example, $b = \sigma_1^3$. Then $b(i) = \sigma_1$ for $i = 1, 2, 3$ and we get

$$\overline{\sigma_1^3} = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \tag{42}$$

which correctly represents the trefoil knot. In the next section we will generalise this example to an infinite family of knots known as torus knots.

3.3 Example: The Torus Knots

The torus knots are an infinity family of prime knots which have particularly simple properties and are frequently used as examples in knot theory texts. The connection between braids and knots is readily illustrated in the case of torus knots and this is what we shall do below.

Definition 3.2. *Given two co-prime (no common factors apart from unity) integers p and q, the torus knot $T_{p,q}$ is constructed by wrapping a closed curve around the surface of a torus such that it encircles it p times meridionally and q times longitudinally (respectively the short and the long way around).*

Torus knots are completely characterized by the two integers p and q. They are invertible (isotopic under switching the orientation) and chiral (*not* isotopic to their mirror image) [?]. The fundamental group of their complements is given by $\pi_1(T_{p,q}) = \langle\{a,b\} : a^p = b^q\rangle$ [49] from which we may recognize the requirement that p and q be co-prime. It may be shown that $T_{p,q} = T_{q,p}$. The torus knots are among the few knots for which the minimal number of crossing-switches required to transform the knot into the unknot, i.e. the *unknotting number*, is known; it is $(p-1)(q-1)/2$ [1]. They are also among the few knots for which the minimal number of crossings in any projection, i.e. the *minimal crossing number* is known; it is $\min[p(q-1), q(p-1)]$ [75].

The simplest example of a torus knot is the trefoil. It is not the unknot even though $T_{p,1}$ for any p would be isotopic to the unknot but p and unity are not coprime. The trefoil knot is $T_{3,2}$ and the knot 5_1 is $T_{5,2}$. In general, the closed p-braid $(\sigma_{p-1}\sigma_{p-2}\cdots\sigma_1)^q$ is isotopic to the torus knot $T_{p,q}$. This is easily seen by picturing $T_{p,q}$ on an actual torus. We cut the torus across a random meridian (the short way around) and straighten it into a cylinder. The remainder will be the braid above. Moreover, there exists no closed braid representative of $T_{p,q}$ with less than p strings so that p is the *braid index* of $T_{p,q}$.

Exercise 3.3. Show that the Alexander polynomial of $T_{p,q}$ is given by

$$\triangle_{T_{p,q}} = \frac{(1-t^{pq})(1-t)}{(1-t^p)(1-t^q)} \tag{43}$$

As we have seen, any torus knot can be represented as the p-braid $b = (\sigma_{p-1}\sigma_{p-2}\cdots\sigma_1)^q$. By using the method of the last section, we can represent $T_{p,q}$ in the polyhedron $P(p-1, q(p-1))$.

$$T_{p,q} = \begin{pmatrix} & & & & 1 & & & & & 1 & \cdots & & & & & 1 \\ & & & \cdots & & & & & \cdots & & \cdots & & & & \cdots & \\ & & 1 & & & & & 1 & & & \cdots & & & 1 & & \\ & 1 & & & & & 1 & & & & \cdots & & 1 & & & \\ 1 & & & & & 1 & & & & & \cdots & 1 & & & & \end{pmatrix} \tag{44}$$

The Dowker code may be obtained from a knot represented in our notation by simply walking through the polyhedron and labelling the crossings. As the polyhedron is structured, this walk is perfectly definite and can be programmed easily on a computer.

3.4 Knots to Braids I: The Vogel Method

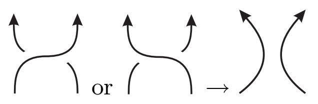

Fig. 17. Both types of crossing have to be reconnected in the shown way in order to obtain a diagram of Seifert circles from a knot diagram.

A braid is more structured than a knot and so the transition from knot to closed braid is harder to effect than the reverse. There exists a simple method due to Vogel [72] which we shall present without proof in this section. Suppose we are faced with a knot diagram D which we want to convert into a closed braid. For this method, we shall have to view D in a variety of ways. From the diagram, we can get to the projection P of the knot onto the plane by viewing each crossing as a double point and thus ignoring over and undercrossing information. From D, we can construct a diagram S by reconnecting each crossing in D in the manner shown in diagram 17. The diagram S will contain a number of unknots which we will call *Seifert circles.* Using these constructions, we can define the crucial concept in Vogel's method.

Definition 3.4 (admissible triple). *Let f be a face of P and a and b be two edges of P. The triple (f, a, b) is called an* admissible triple *if and only if it satisfies: (i) a and b are contained in different Seifert circles and (ii) a and b have the same orientation with respect to any orientation of ∂f, the boundary of f.*

It is shown in [72] that the following algorithm will transform any knot diagram D into a diagram of a closed braid.

Algorithm 3.5 *Input: A knot diagram D. Output: A knot diagram D' ambient isotopic to D and in the form of a closed braid.*

1. Determine if D has an admissible triple. If yes, continue. If no, D is in the form of a closed braid and the algorithm is done.

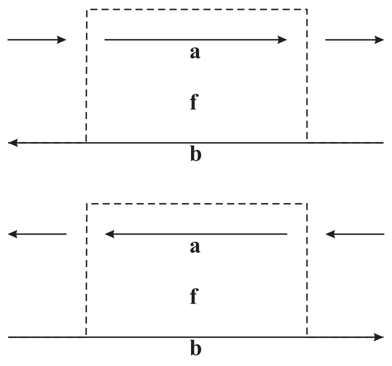

Fig. 18. The two types of admissible triples are shown in the solid curves and the form into which they should be transformed is shown in the dashed curves.

2. Admissible triples can come in the two flavours shown in the solid curves in figure 18. Each admissible triple detected, is to be transformed (via a Reidemeister move type 1) from the solid curve to the dashed curve in figure 18. Such a transformation will be called an *elementary transformation.* Then go back to step 1 of the algorithm.

It can be shown that algorithm 3.5 always terminates after at most $(s-1)(s-2)/2$ elementary transformations where s is the number of Seifert circles in S. The braid which this algorithm generates has at most $n+(s-1)(s-2)$ crossings where n is the number of crossings in D as the elementary transformation adds two crossings each time it is applied. It is unclear whether algorithm 3.5 is confluent, that is whether the order in which we perform elementary transformations should two (possibly overlapping) triples be simultaneously admissible changes the final outcome.

3.5 Knots to Braids II: An Axis for the Universal Polyhedron

Having constructed a new notation for knots, we wish to solve the problem of how to extract a closed braid from the matrix which is isotopic to the knot described by the matrix. A few algorithms have been constructed in the past, which convert a knot into a closed braid but they are difficult to implement because they depend upon topological deformation of the knot projection [51] [15]. The best known algorithms have been implemented [72] [77] and have complexity $O(n^2)$. We shall present an algorithm which achieves the conversion with complexity $O(n)$, increases the number of crossings only in a few cases (and then only by a few crossings) and uses a linearly bounded number of strings. There exists no algorithm to calculate the number of strings which are at least necessary to describe a specific knot — the braid index of the knot. Because of this, it is not possible to say how close to the minimum the number of strings used by our algorithm is. The number of crossings is sometimes increased because it has been found that there are knots for which any closed braid representative has more crossings than the minimal knot diagram; the knot 5.1 in the standard tables is the simplest example of this [66]. Our algorithm is valid both for oriented and unoriented knots.

An Example Alexander's theorem was proven by showing that every knot can be deformed into a form where the knot loops around an axis a finite number of times without local maxima or minima with respect to that axis. If we cut the string along the axis in one place, we obtain a braid. The gluing back of the cut constitutes the canonical closure. Thus as far as the canonical closure is concerned, the finding of an appropriate axis is the key. Having obtained a canonically closed braid which is equivalent to a knot, we may obtain a plait from it by considering the closure curves part of the braid diagram and moving them into the middle of the braid diagram. The next section gives an example of this.

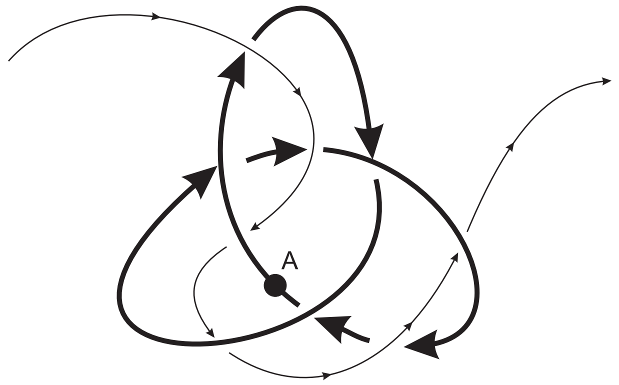

Fig. 19. The trefoil knot with an axis for braiding it.

For the rest of this section, we are going to work through an example of our method. Consider the trefoil knot in figure 19. We have drawn an axis through it by the following method: (1) We drew a line through the projection of the trefoil which intersects every region of the plane at least once, (2) begins and ends in the infinite region and then (3) assigned the under and overpasses of the knot under and over the axis by traversing the knot from a random starting point (point A in the figure) while (4) assigning the passes alternately as we met the crossings of axis and knot. Next we perform a coordinate transformation from the knot reference frame (figure 19) to the axis reference frame in figure 20 by pulling the axis straight.

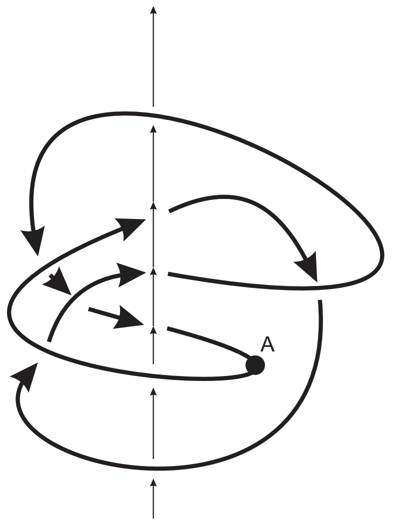

Fig. 20. The trefoil knot as it appears after the axis has been straightened from figure 19. For reference the point A has been labeled here again.

We can easily observe from figure 20 that the axis is valid; i.e. if we traverse the knot starting at A we will travel around the axis without local maxima or minima permanently in a clockwise direction. If we now cut the knot at those points at which it overcrosses the axis and lay out the ends carefully to either side, we shall obtain the braid $\sigma_1^{-1}\sigma_2^{-1}\sigma_1^{-1}\sigma_2^{-1}$ shown in figure 21 (a). To get back to the trefoil from this, we perform the canonical closure which is identical to sealing the cuts made above. This is shown in figure 21 (b). This knot has four crossings and is ambient isotopic to the trefoil thus there is some inefficiency in our braid representation (note however that there exist knots for which the most efficient braid representation contains more crossings than their most efficient knot projection [66]). We note that we may lift the arc labeled in figure 21 (b) to remove one crossing. This move also removes a string and so we obtain the braid of figure 21 (c). This braid has two strings and three crossings, it is thus the most efficient representation of the trefoil as the trefoil must have at least this many strings and crossings. We conclude that the closure of the braid $\sigma_1^{-1}\sigma_1^{-1}\sigma_1^{-1}$ is ambient isotopic to the trefoil knot. Note that we may turn the entire figure 21 (c) about a vertical axis through its center and thus obtain the result that the braid $\sigma_1\sigma_1\sigma_1$ is ambient isotopic to the trefoil also; this, finally, is the well-known braid representation of the trefoil knot. This is the prototype for a general method which we shall develop below.

Platting a Knot The diagram of a knot which is expressed as a closed braid may be naturally divided into two parts: the braid and the closure. The most

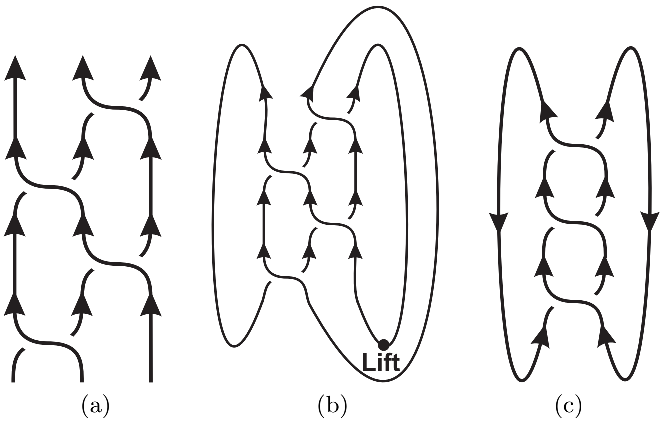

Fig. 21. The braid which is extracted from figure 20 by cutting the trefoil knot at its overcrossings over the axis and laying out the ends is displayed in part (a). The closure of this braid is part (b). If we lift the arc labeled in part (b) we obtain the braid in part (c). See discussion in the text.

important feature of the braid part, for our purpose, is the requirement that all strings be monotonic increasing in the vertical coordinate, that is they may only go side to side and never double back on themselves. In this light, consider turning the polyhedron $P(i,j)$ clockwise by $\pi/2$. If the polyhedron does not contain any ∞ tangles, this is already a canonically closed braid. However, in general, the polyhedron will contain ∞ tangles. Note that the rotation will make the ∞ tangles look like 0 tangles. In an effort to rid ourselves of the ∞ tangles, we take the top string in the ∞ tangle and move it all the way to the bottom of the knot diagram and move the bottom string all the way to the top. In this way, we have created two extra strings in the braid which are closed in the plait manner. If we do this for all ∞ tangles, we will have a valid braid in the center of the diagram but the closure mechanism will be a hybrid between the canonical and plait methods. In order to rectify the situation, we move the strings which are closed in a canonical manner into the center of the braid diagram, thereby creating more strings and more crossings. Once this has been done, we have a fully valid braid closed in the plait manner which is ambient isotopic to the knot we started with. Figure 22 shows the process of converting the unknot

$$U = \begin{pmatrix} -1 & 1 \\ \infty & -1 \end{pmatrix} \tag{45}$$

into the braid $\sigma_2\sigma_4^{-1}\sigma_3\sigma_4\sigma_5^{-1}\sigma_6^{-1}\sigma_4^{-1}\sigma_5^{-1}\sigma_4\sigma_6$ closed in the plait manner. This procedure is valid generally and clearly represents a readily implementable algorithm for transforming a knot given in our notation into a plait. If the original knot is given in the polyhedron $P(i,j)$ and has k tangles of type ∞,

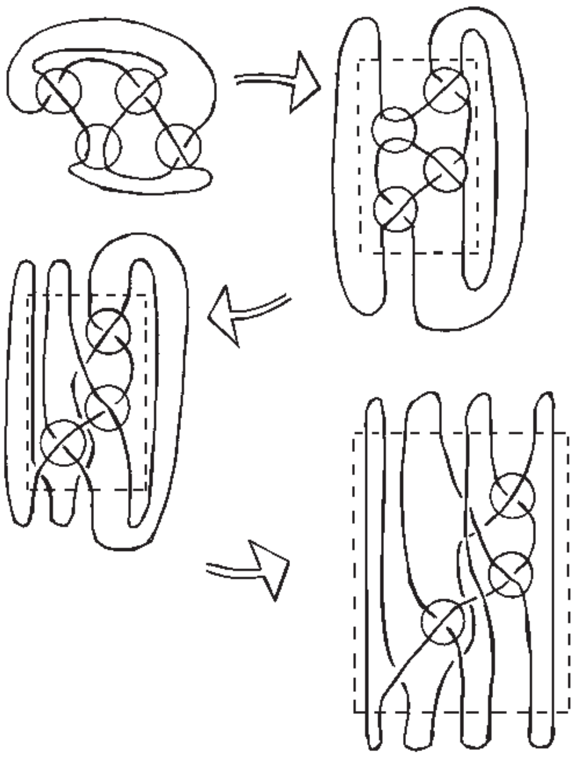

Fig. 22. The conversion of a knot into a plait.

then the number of strings required in the plait is $2(i+k+1)$ but the number of crossings depends upon the exact configuration.

Laying the Axis As mentioned before, the transformation of a knot projection into a canonically closed braid centers around finding an appropriate axis for the string to wind around. This was the central point of Alexander's theorem which proves that such an axis may always be found. A ready method for finding an axis is given in the following algorithm.

Algorithm 3.6 *Input: A knot projection. Output: A knot projection with an axis around which the knot winds without local maxima or minima.*

1. Begin with enumerating the regions into which the knot projection divides the plane, suppose there are R of these.
2. Choose two arbitrary points in the infinite region and call them A and B.
3. Draw a line L connecting A and B in such a way that the line intersects every region at least once.

4. Choose a random point on each of the knot's components and traverse the knot in the direction of the orientation once for each component starting at the chosen point. While traversing label each intersection of L with the knot alternatingly with a $+$ or $-$ sign starting with $+$.
5. Interpret each $+$ crossing as an overcrossing of L over the knot and each $-$ crossing as an undercrossing of L under the knot. The line L oriented from A to B is then a valid axis.

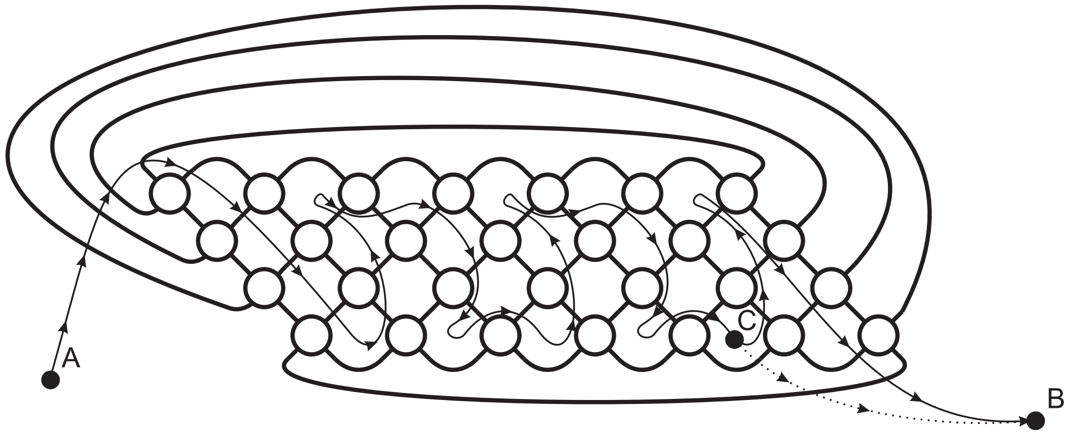

Fig. 23. The axis of the braid through the polyhedron $P(i, j)$.

This algorithm may clearly be applied to our polyhedron $P(i, j)$. However we have the problem of the regions which depends upon the exact configuration of the knot. This can be solved by forcing the line L to intersect every region in the *polyhedron* and therefore intersecting some regions of the *knot* more than once. This is unfortunate but unavoidable if we are seeking a general solution of the problem. The manner in which this may be done most economically is illustrated in figure 23. The line L is the dotted line beginning at point A and finishing at point B. If the polyhedron has an odd number of columns (as the one in figure 23), then the line L is best described by the dotted line in figure 23. If however, the polyhedron has an even number of columns, then the line L is best described by the dotted line in figure 23 from point A to point C and then the dashed line from point C to point B. If algorithm 3.6 is correct then a line drawn in a general polyhedron $P(i, j)$ according to this example is a valid braiding axis.

We may find an axis which passes through every region exactly once, if possible, by the following algorithm.

Algorithm 3.7 *Input: A matrix describing a knot in our notation. Output: A matrix describing the regions of the knot. Each element of the matrix*

receives a label from 1 to R, the number of regions. This gives complete information about which regions of the polyhedron are connected and how many there are.

1. Begin at the top left of vertex $(1, 1)$ and follow the boundary downwards, as for counting regions, the orientation of the knot does not matter. Mark the region $(0, 1)$ with a 1, the current marker, in the region matrix.
2. In following the boundary, one will come to vertex $(1, 1)$; we assess its value and continue. If we stay in the same region of the *polyhedron* we continue, if we enter a new region of the polyhedron, then this new region of the polyhedron belongs to the same region of the *knot* as the previous one and thus we mark it with the current marker in the region matrix. The whole issue at hand is that the regions of the polyhedron are known while we wish to gain knowledge of the regions of the knot.
3. We continue to follow the boundary until we reach the point of origin.
4. We search the matrix for an unmarked region. If there exist unmarked regions, we increment our current marker and choose one of the regions as our new starting region and choose a point upon its boundary as our new starting point. Then, we repeat the algorithm from step 1, marking the region with the current marker.
5. Once no unmarked region of the polyhedron exists, the algorithm is finished. The largest marker used in the matrix which we have obtained is clearly the number of regions of the knot. Furthermore, since all connected regions are labeled with the same marker, we have a complete knowledge of which regions of the polyhedron belong to the same region of the knot.

Algorithm 3.8 *Input: A knot projection given in our notation. Output: An axis which passes through every region exactly once, if this is possible. If not the output is an axis which passes through each region at least once.*

1. Get the region information as prescribed in algorithm 3.7.
2. Construct a graph in which each region is symbolized by a node and two nodes are connected by an unweighed edge if they are adjacent in the plane.
3. A Hamiltonian circuit is then a path which passes through each region, that is node, exactly once starting in the infinite region and returning there. If a Hamiltonian circuit exists, so does an optimal axis. If no Hamiltonian circuit exists, we find an axis using algorithm 3.6 which gives an axis which passes through every region at least once.

The advantage is that we will generate a braid with less strings but the Hamiltonian circuit problem is NP-complete and so the execution of algorithm 3.8 is exponential (unless we use an approximation algorithm or it is shown that P = NP). This fact lends further weight towards the usefulness of algorithm 3.6. The primary usefulness of this algorithm originates in the fact that the laying of the axis does not depend upon the exact knot configuration, only the labelling does. Before we continue, we prove that algorithm 3.6 always yields a valid axis, this essentially amounts to proving Alexander's theorem.

Theorem 3.9. *Given any knot projection, algorithm 3.6 will find an axis about which the knot is without local maxima or minima.*

Proof. Alexander's theorem [4] states given a knot projection, it is possible to deform it with respect to a point P in the projection plane that after the deformation a point A which travels along the knot in the direction of its orientation will travel around the axis defined by P (the axis is a line perpendicular to the projection plane intersecting it at P) in a constant fashion, either clockwise or counterclockwise, for the entire circumnavigation of the knot. We wish to do the opposite, namely to deform the axis around the knot projection to achieve the same ends. We can imagine the process of laying the axis as akin to sewing in which we move the needle up from and down onto the plane. Morton [59] has constructed a similar method to ours which he calls "threading."

The knot divides the plane into several regions. If the axis does not intersect a particular region, the point A will change course during traversing the knot and so the axis must intersect each region. It is however clearly only necessary for the axis to intersect the region once. Choose a line in the plane which intersects the axis. With respect to this line we can define an angular coordinate θ going around the axis. As point A must travel around the axis in a constant fashion it must, after it passes $\theta = 0$, reach $\theta = \pi$ before it once again reaches $\theta = 0$. This shows that the axis, in the projection plane, must over and undercross the knot alternately with respect to A. This fulfills the requirements of an axis and these are assured by algorithm 3.6 and thus the theorem is proven. $\square$

Getting the Braid Having obtained the axis, we must now simply put together all the pieces and construct the braid. This will be done via the following algorithm.

Algorithm 3.10 *Input: An axis L in a knot projection given in $P(i, j)$ using our notation. Output: A braid the canonical closure of which is ambient isotopic to the given knot.*

1. Consider an empty polyhedron $P(i, j)$ and label each edge by the row and column index of the vertex out of which it is emerging on the right side giving it the further label a if it is the top edge and b if it is the bottom edge. That is the top right hand edge coming out of the vertex $(1, 1)$ would be $(1, 1)_a$.
2. All edges which intersect the axis L at a positive crossing are to be numbered in order starting at point A; suppose there are k of these.

3. Starting at the numbered edges, use the traversal algorithm to follow each edge around the knot until another positive crossing with the axis L. All edges encountered are to be labelled with the same number as the original edge.
4. When all edges are numbered, we have identified the individual strings of the braid and numbered them in order. Assign a distance value of 1 to each edge in the polyhedron.
5. Traverse the knot again as in step 3 but this time stopping at each double point and extracting which labelled string passes over which other labelled string and at which distance value this occurs.
6. When the whole has been traversed, we have a list of crossings specifying which strings are involved, which string crosses over the other and at what distance from the bottom of the braid the crossing occurs. This information may be used readily to construct a colored braid, which may be converted easily into an Artin braid word.
7. We assess the string labels around the knot and calculate the permutation associated with the braid which winds around our axis. If this permutation is different from the permutation of the braid which we obtained in step 6, the residual permutation must be added to this braid in the form of extra crossings.

The number of crossings is increased in some circumstances by a small amount in step 7 of the algorithm. It is a fact that there exist knots of minimal crossing number n which have closed braid representatives all of which have crossing numbers greater than n [66]. Hence, step 7 is not a deficiency of the algorithm 3.10 but a fundamental necessity.

It is clear from Alexander's theorem[4] that this algorithm works. The number of strings used is the number of positive crossings of the axis with the knot which is equal to half the number of crossings. The number of crossings of the axis with the knot is

$$N_c = \begin{cases} 4i + (2i+2)\left\lfloor \frac{j-2}{2} \right\rfloor & j \text{ odd} \\ 2i + (2i+2)\left(\frac{j-2}{2}\right) + 2 & j \text{ even} \end{cases} \tag{46}$$

where $\lfloor x \rfloor$ is the greatest integer less than x. An analysis of the possibilities in oddness and evenness of i and j reveals that N_c is always even which is good since we must have an equal number of positive and negative crossings.

Algorithm 3.10 therefore finds a braid with a number of strings which scales linearly in the number of rows and columns necessary to represent the knot. It is conceivable that a more economical way of laying an axis may be found using algorithm 3.8 but this has an exponential complexity. The number of strings may be reduced after the braid has been found using Markov's theorem.

The determination of the regions, the laying of the axis, the labelling of the axis crossings, the labelling of the edges and the extraction of the double point information all take a time proportional to the number of vertices in the polyhedron ij. The building of the braid from the crossing information takes time proportional to ij. Therefore the entire algorithm to proceed from a knot projection to a canonically closed braid has complexity $O(ij)$. This algorithm succeeds in being readily implementable and in constructing a braid which is reasonably small.

3.6 Peripheral Group Systems of Closed Braids

In this section we will investigate the peripheral group system of the closure of the fundamental braids. The peripheral group system is a complete invariant of knots and figures largely in knot theory. The fundamental braid words are very important in braid theory and we choose them for this investigation for that reason and that the closure of Δ_3 is the Hopf Link and the closures of the other fundamental braids look very similar to Hopf Links. In fact so similar that we can regard the class of knots defined by the closure of fundamental braids as a generalization of the Hopf Link. Another generalization of the Hopf link has been investigated in the literature [28]. We shall see that our methods developed here can be extended beyond fundamental braid words to all braid words.

The Fundamental Group Consider a space X and a point $x_0 \in X$ which we shall call the *base point*. In the space X, we may construct loops, i.e. paths from x_0 to itself.

Definition 3.11 (fundamental group). *The group that consists of the loops at x_0 in the space X with respect to the homotopy equivalence relation (continuous maps from one set of loops to another) is called the* fundamental group *of the space X and is denoted by $\pi_1(X, x_0)$.*

It can be shown that if X is path connected, the choice of base point does not matter. All spaces we are about to consider are path connected and so we shall drop the specification of the base point and denote a fundamental group by $\pi_1(X)$.

Definition 3.12 (knot complement). *The* complement *of a knot K with respect to a space X is the space $X - V(K)$ where $V(K)$ represents a tubular neighborhood of the knot K.*

Definition 3.13 (knot group). *The* group *of a knot K, denoted by $\pi(K)$ is the fundamental group of the complement of the knot with respect to the space $X = \mathbb{R}^3$ (sometimes X is taken as S^3 but it can be shown that the two fundamental groups arising from $X = \mathbb{R}^3$ and $X = S^3$ are isomorphic).*

Consider a knot K and its complement $\mathbb{R}^3 - V(K)$. In this space, choose a point x_0 as the base point (the choice is arbitrary as the space is path connected). Consider the projecting cylinder $Z \in \mathbb{R}^3$ which contains all of

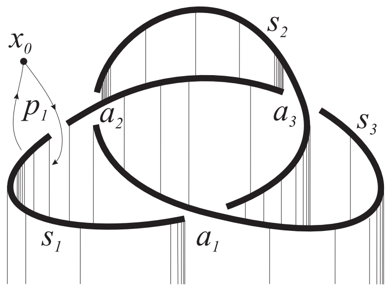

Fig. 24. The labelling of the trefoil knot in order to yield a presentation for the fundamental group of its complement.

$V(K)$ and is constructed such that the projection of K onto the plane $z = 0$ in $\mathbb{R}^3$ contains at most double points. This projecting cylinder will contain n self-intersections if K has n crossings. Label these self-intersections by a_i and the sections of Z between the a_i by s_i. Z is to be oriented in order to match the orientation of K. Denote a loop from the base point to itself which goes exactly once around s_i and no other s_j by p_i. From this construction, it is clear that all loops (with respect to homotopy equivalence) can be constructed as products of the loops p_i. This can easily be seen by the fact that any path in the complement can be continuously deformed into a product of loops p_i. Thus $\pi(K)$ is generated by the loops p_i. For an example see figure 24.

Having gotten the generators, we need the relations to obtain a group presentation. Consider the loops encircling the a_i and join them to the base point by a path c_i, then it is clear that $c_i a_i c_i^{-1}$ is contractible (homotopic to the trivial loop). The word in the group corresponding to this loop is then a relation in the group. The presentation so obtained is called the *Wirtinger presentation* of the knot group. This discussion proves that the following algorithm to obtain it is correct.

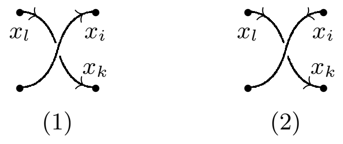

Fig. 25. The two possible forms of double points in the diagram of an oriented knot. The crossing of type 1 has characteristic *epsilon* = 1 and the type 2 has characteristic $\epsilon = -1$.

Algorithm 3.14 (Wirtinger presentation of the knot group) *Input: A knot projection of a knot K. Output: The Wirtinger presentation of $\pi(K)$.*

1. *Label all overcrossing arcs in the projection by g_i for $1 \leq i \leq n$ where n is the number of double points in the projection starting at any point and then assigning the labels in order corresponding to the orientation of the knot.*
2. *For each double point, determine its characteristic sign, see figure 25. The characteristic can be most easily determined by the "right hand rule" which says that if you spread your thumb at right angles from your fingers and point it along the overcrossing arc, and if your finger point along the undercrossing arc, the characteristic is 1 and -1 otherwise (along being taken to mean along the orientation of the knot).*
3. *$\pi(K) = \langle\{g_1, g_2, \cdots, g_n\} \,|\, \{r_1, r_2, \cdots, r_n\}\rangle$ where the relators are given by $r_j = g_j g_i^{-\epsilon_j} g_k^{-1} g_i^{\epsilon_j}$ and ϵ_j is the characteristic of the crossing associated with r_j.*

It can be seen that one relation can be derived from the others and thus a knot group has deficiency one. The knot group is an invariant of knots as is trivially seen by definition but it is not complete. Thus if two knots have isomorphic knot groups they are not necessarily isotopic. However if they have non-isomorphic groups, the knots are distinct. The unknot U has the knot group constructed by a single generator and no relations, i.e. $\pi(U) = \mathbb{Z}$, the infinite cyclic group. It is a practical observation that the Wirtinger presentation can often be simplified considerably in that some generators are removable [38]. In particular, $\pi(K)$ for the torus knot $T_{p,q}$, which is a knot that winds around a torus p times the short way around (meridionally) and q times the long way around (longitudinally), is given by $\pi(K) = \langle\{a, b\} : a^p = b^q\rangle$ [49].

The Square and Granny Knots The knot group is not a complete invariant. We wish to illustrate this by an example. The knot groups of the square S and granny G knots are isomorphic but the knots are distinct. The demonstration of this fact will occupy this section. For a picture of these knots, see figure 26. We follow algorithm 3.14 to compute the knot groups. Both knots receive six generators and six relations. All crossings in the granny knot have characteristic -1 as well as three crossings in the square knot, the other crossings in the square knot have characteristic 1. We thus write down the Wirtinger presentations.

$$\pi(G) = \left\langle \begin{array}{c} \{g_1, g_2, g_3, g_4, g_5, g_6\} \,|\, g_6 g_2 = g_2 g_1;\; g_5 g_1 = g_1 g_6; \\ g_1 g_6 = g_6 g_2; g_2 g_4 = g_4 g_3;\; g_4 g_3 = g_3 g_5;\; g_3 g_5 = g_5 g_4 \end{array} \right\rangle \tag{47}$$

$$\pi(S) = \left\langle \begin{array}{c} \{g_1, g_2, g_3, g_4, g_5, g_6\} \,|\, g_6 g_2 = g_2 g_1;\; g_5 g_1 = g_1 g_6; \\ g_1 g_6 = g_6 g_2; g_3 g_2 = g_2 g_4;\; g_2 g_4 = g_4 g_3;\; g_4 g_3 = g_3 g_5 \end{array} \right\rangle \tag{48}$$

Three generators may be defined in terms of the other three in both groups and after some simple manipulation we obtain

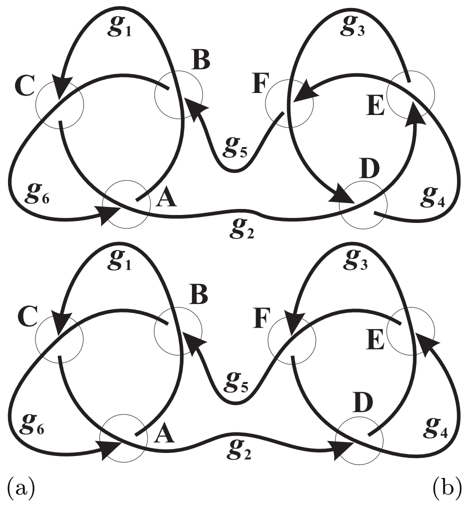

Fig. 26. The construction of the Wirtinger presentation of the fundamental group of the complement of the (a) square and (b) granny knots.

$$\pi(G) \approx \pi(S) = \left\langle \begin{array}{l} \{g_1, g_2, g_3\} \mid \\ g_1 g_2 g_1 = g_2 g_1 g_2;\ g_2 g_3 g_2 = g_3 g_2 g_3 \end{array} \right\rangle \tag{49}$$

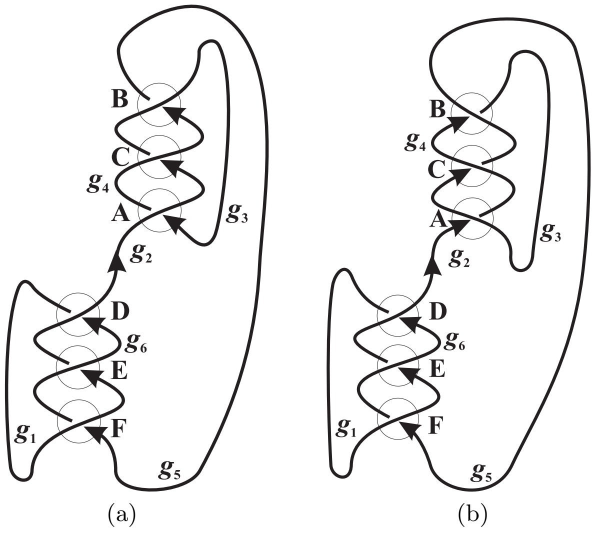

Fig. 27. The (a) square and (b) granny knots from figure 26 transformed into closed braids.

Even though the knot groups are isomorphic, the knots are distinct. This can seen by deforming the knots into closed braids and computing their Jones polynomials. The transformation is straightforward and the result is shown in figure 27. Reading off from the figure, we obtain that

$$G \approx \overline{\sigma_1^{-3}\sigma_2^3}; \qquad S \approx \overline{\sigma_1^3\sigma_2^3} \tag{50}$$

Recalling the algorithm to compute the Jones polynomial yields

$$V_G t = 3 - \left(t^3 + t^{-3}\right) + \left(t^2 + t^{-2}\right) - \left(t + t^{-1}\right); \qquad V_S(t) = \left(t + t^3 - t^4\right)^2 \tag{51}$$

and we see that the knots are distinct proving that the fundamental group is a incomplete invariant.

Peripheral Group System The fundamental group is not a complete invariant but it is possible to refine our methods to construct a complete invariant from the fundamental group, the peripheral group system. This is the only complete invariant of knots that we can readily compute for all knots. The problem is that distinguishing knots via the peripheral group system requires distinguishing groups with respect to isomorphism which is known to be an undecidable problem [2, 3, 64].

The complement of a knot is uniquely specified (up to isomorphism) by its peripheral group system which consists of the fundamental group and a few subgroups thereof (this is Waldhausen's theorem [73], see [40] for a more accessible proof). It is however known that the word problem for any fundamental group of any knot is solvable [74]. If the knot is alternating, the conjugacy problem is also solvable [5].

We define the *linking number* of two curves a and b, denoted by $lk(a,b)$ as the weighted sum of the characteristics ϵ of each crossing. We define a *meridian* m_i and a *longitude* l_i of a knot component K_i by requiring the following properties: (1) m_i and l_i are oriented, polygonal, simple and closed curves in $\partial V\left(K_i\right)$, the boundary of the thickened neighborhood of K_i which we denote by $V\left(K_i\right)$, (2) m_i and l_i intersect in exactly one point, (3) m_i is null homologous ($m_i \sim 0$) in $V\left(K_i\right)$ and $l_i \sim K_i$ in $V\left(K_i\right)$, (4) $l_i \sim 0$ in $C\left(K_i\right)$ and (5) $lk\left(m_i, K_i\right) = 1$ and $lk\left(l_i, K_i\right) = 0$ in S^3. The above five properties define m_i and l_i uniquely up to isotopy on the boundary of $V(K)$ [**?**] (see figure 28 for an illustration). The *meridian-longitude system pair* $\mathcal{M}(K)$ for a j-component knot K is the pair of sets $\left(\{m_1, m_2, \cdots, m_j\}, \{l_1, l_2, \cdots, l_j\}\right)$. Interestingly, if the longitude is trivial (equivalent to the identity element) in the knot group, the group is infinite cyclic; i.e. the group is isomorphic to the fundamental group of the complement of the unknot. Furthermore, the meridians and longitudes commute with each other in any knot group [49].

The meridians can be taken as the Wirtinger generators of the knot group. The longitude of a knot may be read off the projection very easily. Begin with the first generator and traverse the knot in the direction of its orientation.

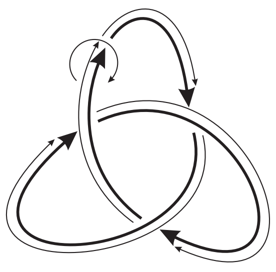

Fig. 28. The thick curve displays the trefoil knot with an orientation. The thin curve which is parallel to the trefoil knot is the longitude; the orientation of the longitude is the same as the knot. The thin curve encircling both trefoil and longitude at the top left hand corner is the meridian. Note that the five conditions given in the text are fulfilled by these curves.

Whenever undercrossing an arc, write down the generator of the undercrossed arc to the power of the negative of the characteristic of the crossing. After the full traversal, append as many copies of the initial generator (or its inverse) to make the total sum of exponents equal to zero.

The meridians and longitudes of $\mathcal{M}(K)$ may be considered to be elements of $\pi(K)$ by choosing a path p_i in $C(K)$ from the base point x_0 to the (unique by definition) point $m_i \cap l_i$ for each i. Then the subgroup $\langle m_i, l_i \rangle$ of $\pi(K)$, generated by m_i and l_i is independent of the choice of p_i up to conjugation. The *peripheral group system* of a j-component knot K is $p(K) = (\pi(K); \mathcal{M}(K))$. By an isomorphism ϕ between two peripheral group systems $p\,(K) \approx_\phi p\,(K')$, we mean $\pi\,(K) \approx \pi\,(K')$ such that $\phi\,(m_i) = m'_i$ and $\phi\,(l_i) = l'_i$ for all i. It can be shown that for any two knots K_1 and K_2, $p\,(K_1) \approx p\,(K_2)$ if and only if $K_1 \approx K_2$ [49]. If we restrict attention to prime knots of a single component, we have $\pi\,(K_1) \approx \pi\,(K_2)$ if and only if $K_1 \approx K_2$ [49]. Thus the problem of knot isotopy can be transformed into the problem of peripheral group system isomorphism. Since it is not possible to determine, in general, if two groups are isomorphic, this does not solve the knot classification problem.

4 Classification of Braids and Knots

In this section we will discuss the word, conjugacy, Markov and minimal word problems introduced in section 2.8 in detail. There are a number of distinct solutions to the word and conjugacy problems in B_n each of which has special features and gives additional insight into the problem and braids in general. We shall discuss briefly the solutions due to Garside as they are historically the most important ones and have had ground-breaking influence on the field. Then we shall discuss a new solution to both problems. The Markov problem is unsolved at present but we will discuss some of its features and why it is so difficult. Lastly, we will present a number of results about the minimal word problem. It is computationally expensive to solve (NP-Complete) and so we present a heuristic algorithm and some simulation approaches and interesting results arising from them.

4.1 The Word Problem I: Garside's Solution

In the Artin presentation, the word problem asks whether two n-braid words $\alpha = \sigma_{a_1}^{\epsilon_1}\sigma_{a_2}^{\epsilon_2}\cdots\sigma_{a_o}^{\epsilon_o}$ and $\beta = \sigma_{b_1}^{\eta_1}\sigma_{b_2}^{\eta_2}\cdots\sigma_{b_p}^{\eta_p}$ such that $1 \leq a_i, b_j < n$ and $\epsilon_i, \eta_j = \pm 1$ for $1 \leq i \leq o$ and $1 \leq j \leq p$ are equivalent to each other (denoted $\alpha \approx \beta$) under the equations $\sigma_i\sigma_j \approx \sigma_j\sigma_i$ for $|i-j| > 1$ and $\sigma_i\sigma_{i+1}\sigma_i \approx \sigma_{i+1}\sigma_i\sigma i + 1$. This problem is traditionally approached by trying to find an algorithm which will use the two braid group relations to construct, from any braid, a unique normal form. A *unique normal form* $\underline{\alpha}$ of the braid α and the corresponding form $\underline{\beta}$ of the braid β are exactly equal ($\underline{\alpha} = \underline{\beta}$) if and only if $\alpha \approx \beta$.

Artin was the first to describe a normal form to solve the word problem for braids [7] but the normal form has a exponentially growing number of crossings in terms of the original number of crossings and so is not as useful as other methods. Garside constructed another unique normal form which has many important properties [37]. Garside's method begins with the crucial proposition,

Proposition 4.1. *For any σ_i^{-1}, we have $\sigma_i^{-1} \approx \Delta_n^{-1}\Delta_{n-1}d_{n-1,i+1}d_{i-1,1}$.*

Together with the fact that $\sigma_i\Delta_n \approx \Delta_n\sigma_{n-i}$ (see proposition 2.19), this means that we may put any n-braid word α into the form $\alpha \approx \Delta_n^{-q}\alpha'$ where α' is a positive braid word and q the number of inverse generators present in α. Next we need to construct the diagram $D(\alpha')$ of the positive braid α'.

Definition 4.2. *The* Cayley diagram $D(b)$ *(or* diagram *for short) of a positive braid word b is the set of all braid words equivalent to b.*

Because of proposition 2.21 which states that two positive and equal braid words are positively equal, the following algorithm obviously constructs $D(b)$ given a positive b.

Algorithm 4.3 *Input: A positive braid word b. Output: The diagram $D(b)$ of b.*

1. We define the set $D_0(b)$ by $D_0(b) = \{b\}$.
2. The set $D_i(b)$ is obtained from the set $D_{i-1}(b)$ by adding to $D_i(b)$ all the braid words which can be obtained from any member of $D_{i-1}(b)$ by applying any braid group relation exactly once and the deleting all those which are already contained in the sets $D_j(b)$ for $0 \leq j < j$.

3. Step 2 is recursively applied until an integer m is reached for which $D_m(b) = \emptyset$. It is obvious from the construction of the sets that a finite m always exists.
4. The set $D(b)$ is then the union of the D_i for $0 \leq i \leq m$.

We note that, in general, $D(b)$ contains a number of elements which grows exponentially both with the length and number of strings in b. Once $D(\alpha)$ has been fully enumerated we select from it a word of the form $\Delta_n^p \alpha''$ for which p is maximal. Then we construct $D(\alpha'')$ and choose the braid word α''' from $D(\alpha'')$ which has the lowest integer associated with it, the decimal expansion of which is given by the concatenation of the generator indices in the braid word ($\sigma_2\sigma_1\sigma_3$ has associated integer 213). The braid word α''' is called the *base* of the diagram $D(\alpha'')$. We define the *Garside normal form* of the braid α to be $\alpha_G = \Delta_n^{p-q}\alpha'''$. We refer to $p - q$ as the *Garside exponent* and to α''' as the *Garside remainder*. It can be shown that the Garside normal form is a unique normal form and that thus two n-braid words α and β satisfy $\alpha \approx \beta$ if and only if $\alpha_G = \beta_G$. Executing this method in practise is time-consuming due to the size of the diagram. There exists an efficient polynomial-time algorithm to extract the maximal number of Δ_n from α' due to Jacquemard [44] which we shall not present here.

The word problem was first solved by Artin [7] and then by Garside [37]. Both of these solutions were algorithmic with exponential complexity. As mentioned above, Garside's algorithm can be made polynomial time due to a new algorithm by Jacquemard [44] to extract braids. The word problem for braids is important enough for many people to have studied it after Garside. The most efficient algorithm (linear in n and quadratic in L) is given in [15]. Below we present a new algorithm based on rewriting systems. It is not as efficient as the best algorithm for the word problem but it is easily generalizable to the conjugacy case and it is very simple to apply. Because of these features, we regard it as a competitive algorithm.

4.2 The Word Problem II: Rewriting Systems

In this section we will develop a new algorithm based on rewriting systems which are used as a tool in theoretical computer science. We begin with a finite alphabet of *constants* $\mathcal{A}$ and a finite set of *variables* $\mathcal{X}$. A *term* t is a finite ordered sequence of constants and variables $t = a_1 a_2 \cdots a_n$ with $n \geq 0$ (i.e. empty terms are allowed) and $a_i \in \mathcal{A} \cup \mathcal{X}$. A *word* w is a finite ordered sequence of constants $w = b_1 b_2 \cdots b_m$ with $m \geq 0$ and $b_i \in \mathcal{A}$. A *substitution* ρ for a term t is a map which assigns a word to each variable in t; the resultant word is denoted by ρt. A *term rewriting system* (TRS) $\mathcal{R} = \{(l_i, r_i)\}$ is a set of ordered pairs of terms l_i and r_i. Each ordered pair in $\mathcal{R}$ is referred to as a *rule* or *rewrite rule* and is often written in the form $l_i \to r_i$; the whole TRS is sometimes denoted by $\to_{\mathcal{R}}$. A TRS $\mathcal{R} = \{(l_i, r_i)\}$ is applied to a word w_0 by determining if w_0 contains the word ρl_i, for some substitution ρ, as a subword. If and only if w_0 contains ρl_i is ρl_i replaced by ρr_i. If $\rightarrow$ is a rewrite rule, then $\leftarrow$ is its inverse, $\leftrightarrow$ is its symmetric closure $(\leftarrow \cup \rightarrow)$ and $\rightarrow^\star$ is its reflexive-transitive closure $(\rightarrow \circ \rightarrow \circ \cdots \circ \rightarrow)$. Two terms t and s are said to be *joinable* if there exists a term r such that $t \rightarrow^\star r \leftarrow^\star s$. Any l_i is called a *redex* and any r_i is called a *reduct* (these are abbreviations of reducible expression and reduced term).

A word w_0 is thus rewritten into a word w_1 if and only if $\mathcal{R}$ may be applied to w_0. We may generate a *rewrite chain* of words $w_0 \rightarrow_\mathcal{R} w_1 \rightarrow_\mathcal{R} \cdots$ in this manner. $\mathcal{R}$ *terminates* if and only if there exists no rewrite chain of infinite length. $\mathcal{R}$ is *locally confluent* if and only if any local divergence $\leftarrow \circ \rightarrow$ is contained in the joinability relation $\rightarrow^\star \circ \leftarrow^\star$. $\mathcal{R}$ is *confluent* if and only if any divergence $\leftarrow^\star \circ \rightarrow^\star$ is contained in the joinability relation $\rightarrow^\star \circ \leftarrow^\star$. $\mathcal{R}$ is *complete* if it is confluent and terminates. If $\mathcal{R}$ is complete a unique normal form exists for each word [8]; the final form obtained by applying $\mathcal{R}$ to the word a maximum number of times.

It should be noted that the computational power of term rewriting systems is identical to that of Turing machines, i.e. one may be simulated by the other [70]. According to the Church-Turing thesis [29], this means that any function which may reasonably be termed computable is computable using a TRS.

It was proven by Birkhoff [24] that the symmetric-reflexive-transitive closure $\leftrightarrow^\star_\mathcal{R}$ of a TRS $\mathcal{R} = \{(l_i, r_i)\}$ is equivalent to the set of equations $\mathcal{E} = \{l_i = r_i\}$. It is an obvious corollary to Birkhoff's theorem that if there exists a complete TRS $\mathcal{R}$ over the alphabet $\mathcal{A} = \{f_i, f_i^{-1}\}$ for which $\leftrightarrow^\star_\mathcal{R}$ contains exactly the equations $\{\mathcal{E}, f_i f_i^{-1} = e, f_i^{-1} f_i = e\}$, then $\mathcal{R}$ solves the word problem for the group $G = \langle \{f_i\}, E \rangle$. Note that $\mathcal{R}$ also solves the word problem for the monoid associated with G, i.e. the monoid obtained when the inverses of the generators are added to the set of generators and the fact that the generators and inverses are in fact inverses $(f_i f_i^{-1} = e, f_i^{-1} f_i = e)$ added to the set of equations.

Termination It is, in general, undecidable whether a TRS terminates or not [43]. Since any Turing machine can be modeled using a TRS this is essentially due to the undecidability of whether a Turing machine will stop, the Turing Halting Problem [71]. It is however decidable for a TRS without any variables [33]. Thus, in general, a termination proof is specific to a particular TRS and must be given for it. A common strategy for proving termination is to use a reduction order on the symbols involved in the TRS. We define a *reduction order* $<_o$ as a strict order over the alphabet and variables of the TRS which satisfies:

1. **compatibility**: For all terms u, v for which $u <_o v$, we have $xuy <_o xvy$ for any terms x and y.
2. **closure**: For all terms u, v for which $u <_o v$ and all substitutions σ, we have $\sigma u <_o \sigma v$.
3. **basis**: $<_o$ is well-founded, i.e. there exists a simplest term under $<_o$.

If one can show that every possible rewriting operation simplifies any term with respect to such an ordering, then the TRS terminates [32]. Furthermore, a TRS $\mathcal{R}$ terminates if and only if there exists a reduction order $<_o$ which satisfies $r_i <_o l_i$ for every rule $l_i \rightarrow r_i \in \mathcal{R}$ [8]. This is true because every step of the rewriting process simplifies the term and there exists a simplest term. Another useful result is that a TRS terminates if and only if it terminates for all instances of its redexes [34]. Some conditions under which the union of two terminating TRSs is terminating are analyzed in [34].

Confluence Like termination, confluence is, in general, undecidable [8]. However, for terminating systems there exists a mechanizable method for deciding confluence [41] that rests on Newman's Lemma which states that a terminating TRS is confluent if and only if it is locally confluent [62] (we shall prove a generalization of this in lemma 4.11). Local confluence can be decided by a systematic method which searches for critical pairs in the TRS. The concept of critical pairs is difficult to trace in history; for an attempt at a historical survey see [27] and for a good technical treatment see [41].

Given a TRS $\mathcal{R} = \{(l_i, r_i)\}$, an *overlap* is a word $w = abc$ such that $ab = \rho l_i$ and $bc = \eta l_j$ for some words a, b and c, two (possibly equal) integers i and j and substitutions ρ and η. Clearly the overlap abc may be rewritten to both $\rho r_i c$ and $a \eta r_j$. An overlap is *non-critical* if the reducts are joinable, $\rho r_i c \leftrightarrow^\star_{\mathcal{R}} a \eta r_j$ and *critical* otherwise. A *critical pair* is the (unordered) pair $(\rho r_i c, a \eta r_j)$ which arises from a critical overlap. It is obvious that if $\mathcal{R}$ contains critical pairs, it can not be confluent. The fact that the non-existence of critical pairs is both a necessary and sufficient condition for local confluence is called the Critical Pair Lemma [46]. Later we shall prove lemma 4.12 which contains the Critical Pair Lemma.

Completeness If we can find a reduction order for a TRS $\mathcal{R}$, thereby prove termination and find that there are no critical pairs, $\mathcal{R}$ is complete and thus solves the word problem for $\leftrightarrow^\star_{\mathcal{R}}$. A general procedure for what to do when we can not do this is called Knuth-Bendix completion from their seminal paper [50]. Again a historical account of this procedure is tangled and [27] is an attempt to unravel it. We shall follow the common practice to call it Knuth-Bendix completion even though, by their own admittion, the initial idea was not theirs.

Suppose we have a set of equations $\mathcal{E}$ on an alphabet $\mathcal{A}$ and a total order $<_{\mathcal{A}}$ (this is a reduction order) on $\mathcal{A}$. Construct a TRS $\mathcal{R}$ from $\mathcal{E}$ by creating a rule $l_i \rightarrow r_i$ in $\mathcal{R}$ from the equation $l_i = r_i$ in $\mathcal{E}$ for all equations in $\mathcal{E}$ such that the rules are ordered such that $l_i >_{\mathcal{A}} r_i$. Now $\leftrightarrow^\star_{\mathcal{R}}$ is equivalent to $\mathcal{E}$ and each rule represents a simplification in terms of $<_{\mathcal{A}}$. Clearly there exists a simplest word, the empty word, and so $\mathcal{R}$ terminates.

If there are no critical pairs, $\mathcal{R}$ is locally confluent and thus complete. If there are critical pairs, order them with respect to $<_{\mathcal{A}}$ and append them to $\mathcal{R}$ as new rules. Termination still holds and so we continue this process.

We may delete duplicate or redundant rules from $\mathcal{R}$ between the steps of this method to obtain a smaller TRS. If this method terminates, we have a complete system [50] [42]. If it does not terminate, a complete system may still exist which contains an infinite number of rules. It is possible to collect an infinite number of rules into a finite number of rules by introducing variables. The Knuth-Bendix algorithm has been implemented by several people and can be used to determine, in some cases, whether a complete system exists. The CiME implementation was used for this thesis [53]. Producing rules with variables and proving the non-existence of critical pairs is, at present, beyond the computer implementations and must be done manually.

The process described here is simplified; there are more pitfalls, in general, and the method has been considerably extended to take into account many other features (many relevant references are in [27]). The method as described is enough for our purposes here however and is generally enough for a word problem in a finitely presented group.

Having reviewed TRS's, we are now in a position to find a TRS for the word problem in B_n. The braid group B_n is defined formally as

$$\begin{aligned} B_n = \langle\, \{\sigma_1, \sigma_2, \cdots, \sigma_{n-1}\} &: \sigma_i\sigma_{i+1}\sigma_i = \sigma_{i+1}\sigma_i\sigma_{i+1}; \\ &\sigma_i\sigma_j = \sigma_j\sigma_i \text{ for } |i-j| > 1\rangle \end{aligned} \tag{52}$$

Given a finitely presented group $G = \langle X, E\rangle$, we can define an associated monoid $M(G) = \left\langle X \cup X^{-1}, E \cup aa^{-1} = 1\right\rangle$ for any $a \in X$. It is clear that the equivalence and conjugacy classes of the group G and the monoid $M(G)$ are identical. In order to solve the word problem for B_n, we augment the monoid $M\left(B_n\right)$ with the generator of the center of B_n, Δ_n^2 to form the monoid

$$\begin{aligned} M^+\left(B_n\right) = \langle\, \{\sigma_1^{\pm1}, \sigma_2^{\pm1}, \cdots, \sigma_{n-1}^{\pm1}, \Delta_n^{\pm2}\} &: \Delta_n^{\pm2}\sigma_i = \sigma_i\Delta_n^{\pm2}; \\ &\Delta_n^{\pm2}\Delta_n^{\mp2} = \sigma_i^{\pm1}\sigma_i^{\mp1} = e; \\ &\sigma_i^{\pm1}\sigma_j^{\pm1/\mp1} = \sigma_j^{\pm1/\mp1}\sigma_i^{\pm1} \text{ for } |i-j| > 1; \\ &\sigma_i^{\pm1}\sigma_{i+1}^{\pm1}\sigma_i^{\pm1} = \sigma_{i+1}^{\pm1}\sigma_i^{\pm1}\sigma_{i+1}^{\pm1}\ \rangle \end{aligned} \tag{53}$$

It is obvious from the definition of the monoid $M^+\left(B_n\right)$ that a solution of its word and conjugacy problems provides a solution for the word and conjugacy problems in the group B_n.

We will use Knuth-Bendix completion upon the oriented rules of $M^+\left(B_n\right)$ under the reduction order $<_b$

$$\Delta_n^2 <_b \Delta_n^{-2} <_b \sigma_1 <_b \sigma_2 <_b \cdots <_b \sigma_{n-1} <_b \sigma_1^{-1} <_b \sigma_2^{-1} <_b \cdots <_b \sigma_{n-1}^{-1} \tag{54}$$

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 subbraid 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^{2k} 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.

$$\begin{aligned}\mathcal{W}_n = \{\, &(1)\ \sigma_i^{-1} \to \prod_{j=1}^{i-1} [d_{j,1} a_{1,j}]\, d_{i,1} a_{1,i-1} \prod_{j=i+1}^{n-1} [d_{j,1} a_{1,j}] \ \ \&\ \ k \to k-1; \\ &(2)\ \sigma_i \sigma_j \to \sigma_j \sigma_i \text{ for } j < i-1; \\ &(3)\ \sigma_i \sigma_{i-1} P \sigma_i \to \sigma_{i-1} \sigma_i \sigma_{i-1} P; \\ &(4)\ \sigma_i \sigma_{i-1} Q \sigma_{i-1} R d_{i,j} \to \sigma_{i-1} \sigma_i \sigma_{i-1} Q d_{i-1,j} \sigma_i R^+ \text{ for } j < i; \\ &(5)\ \prod_{i=1}^{n-1} d_{i,1} a_{1,i} S_i \to \prod_{i=1}^{n-1} S_i \ \ \&\ \ k \to k+1 \,\}\end{aligned} \tag{55}$$

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. A similar, unpublished TRS was also found using Knuth-Bendix completion by Yoder [78]. Rules 1 and 5 are simple to understand; the other rules are illustrated in figure 29.

Theorem 4.4. *$\mathcal{W}_n$ is complete and solves the word problem for B_n.*

Proof. It can be checked easily but laboriously that the system terminates, there are no critical pairs and that its symmetric-transitive closure is the monoid we began with which proves the theorem.

The rules of a TRS are to be applied in a non-deterministic way and a complete TRS always reaches the unique normal form no matter what strategy of rule application is chosen [8]. Since $\mathcal{W}_n$ is complete and all strategies are equivalent, we will choose the following strategy.

Algorithm 4.5 *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.

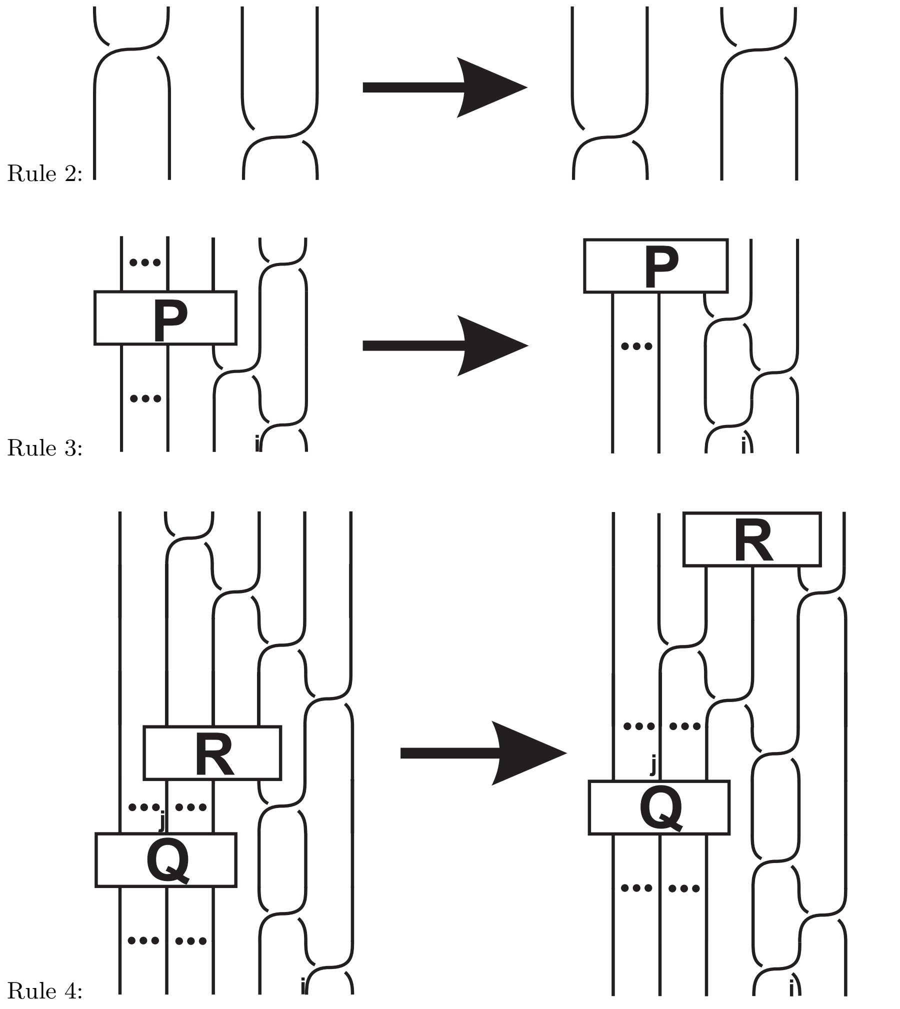

Fig. 29. Rules 2, 3 and 4 of TRS $\mathcal{W}_n$ illustrated.

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 4.5 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 4.6. *$\mathcal{W}_n$ solves the word problem for any word $w \in B_n$ of word length l with complexity $O\left(l^2 n^4\right)$.*

Proof. This can be checked easily by counting how many times the rules are used.

4.3 The Conjugacy Problem I: Garside's Solution

Extending the word problem solution of Garside to the conjugacy case is not hard. First we define two new terms.

Definition 4.7. *If a positive word* $b = if$ *for two braid words* i *and* f *such that* $1 \leq L(i) \leq L(b)$, *then* i *is called an* initial route *of* b *and* f *an* associated final route. *Note that the definition requires exact equality* $(=)$ *and not group theoretic equivalence* $(\approx)$.

We begin by constructing all possible initial routes of Δ_n. This is easily done using $D(\Delta_n)$. Replace all of these routes by the base of their diagrams and delete duplicates. This is the *set of initial routes* of Δ_n and denoted by $I(\Delta_n)$.

We will now form a set $S(\beta)$ for a braid word β called the *summit set* of the braid. This set is the analogue of the Cayley diagram, which we used for the word problem, for the conjugacy problem. We construct the summit set as follows. First construct the Garside normal form of β, i.e. β_G, and note its exponent. The set $S_1(\beta)$ is then constructed by conjugating β_G by each word of $I(\Delta_n)$, computing the Garside normal form of the result and deleting all those words of lower exponent than β_G. Iteratively, the set $S_i(\beta)$ is obtained from the set $S_{i-1}(\beta)$ by conjugating each element of $S_{i-1}(\beta)$ by each word of $I(\Delta_n)$, computing the Garside normal form of the result and deleting all those words of lower exponent than β_G.

Suppose the Garside exponent of β_G is m and its exponent sum is s. Every word in any $S_i(\beta)$ has exponent sum s and Garside exponent $p \geq m$ by definition. The Garside remainder of β_G is $\underline{\beta_G}$ and so we have

$$s = L\left(\underline{\beta_G}\right) - mL(\Delta_n) \tag{56}$$

$$m = \frac{L\left(\underline{\beta_G}\right) - s}{L(\Delta_n)} \tag{57}$$

$$p \leq \frac{s - L\left(\underline{\beta_G}\right)}{L(\Delta_n)} \tag{58}$$

but we also have $p \geq m$ and so there are only finitely many p that satisfy the requirements. Hence there are only finitely many distinct words with Garside exponent p and exponent sum s. Thus the process of constructing the sets $S_i(\beta)$ terminates, i.e. there exists a finite integer j such that $S_j(\beta) = S_{j+1}(\beta)$.

Definition 4.8. *We define the summit set* $S(\beta)$ *to consist of all the elements of* $S_j(\beta)$ *which have maximal Garside exponent. The* summit exponent t *is the Garside exponent of all these braid words and the* summit remainder *is* *the Garside remainder $\underline{\beta_S}$ of the unique element in $S(\beta)$ with smallest tail. The* summit form *of β is then $\beta_S = \Delta_n^t \underline{\beta_S}$.*

We state, without proof, the main result of Garside.

Theorem 4.9 (Garside's Conjugacy Theorem [37]). *Two braid words $\beta_1, \beta_2 \in B_n$ are conjugate if and only if their summit forms are identical.*

Proof. For a proof, see [37, 14].

Like the word problem, the conjugacy problem is very important for braid and knot theory and so has received much attention. The first solution was produced by Garside [37] and the best algorithms are given in [15, 36]. We should remark that Jacquemard [44] has used his extraction algorithm to obtain good practical results for small n. All these algorithms still require an exponential amount of computation time as a function of n and L. The important question of whether conjugacy is solvable in polynomial time is solved positively in the next section.

4.4 The Conjugacy Problem II: Rewriting Systems

Conjugacy in Free Groups 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 [45]. A general word $w \in \mathcal{F}_n$ takes the form

$$w = f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_m}^{p_m}, \qquad 1 \leq s_k \leq n \tag{59}$$

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

$$\mathcal{R}_w(\mathcal{F}_n) = \{f_s^p f_s^q \to f_s^{p+q},\ \forall 1 \leq s \leq n\} \tag{60}$$

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 the length of the word 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 (59) by

$$C^i(w) = f_{s_j}^{p'_j} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_j}^{p''_j} \tag{61}$$

such that

$$p'_j + \sum_{k=j+1}^{m} p_k = i \tag{62}$$

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 w one by one. We shall say that two words w and w' are *cyclicly permutable* (denoted $\approx_{cp}$) if and only if there exist an i such that $C^i(w) \approx w'$. It is obvious that cyclic permutability forms an equivalence relation for any group G.

Proposition 4.10. *For any group G, the equivalence relation of cyclic permutability ($\approx_{cp}$) is identical to that of conjugacy ($\approx_c$).*

Proof. Any group G has a presentation which may be obtained from some free group $\mathcal{F}_n$ of rank n by adding relations [31]. Moreover, if the conjugacy problem is solvable in one representation, it is solvable in all [57]. Suppose $w \approx_{cp} w'$, then there exists an i for which

$$w' \approx C^i(w) = f_{s_j}^{p'_j} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_j}^{p''_j} \tag{63}$$

where

$$p'_j + \sum_{k=j+1}^{m} p_k = i \tag{64}$$

Let

$$\gamma = f_{s_1}^{p_1} f_{s_2}^{p_2} \cdots f_{s_j}^{p''_j} \tag{65}$$

Then

$$w' \approx f_{s_j}^{p'_j} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} \gamma \tag{66}$$

$$\approx \gamma^{-1} \gamma f_{s_j}^{p'_j} \cdots f_{s_{m-1}}^{p_{m-1}} f_{s_m}^{p_m} \gamma \tag{67}$$

$$\approx \gamma^{-1} w \gamma \tag{68}$$

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

$$w' \approx \gamma^{-1} w \gamma \tag{69}$$

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

$$C^{L(\gamma)}(w') \approx \gamma \gamma^{-1} w \approx w \tag{70}$$

Thus $w \approx_{cp} w'$. □

We will refer to the set of words which contains the word w and all its cyclic permutations as the *cyclic word* $c(w)$. If $L(w) = m$, then this set contains $|c(w)| = m$ elements. Given two cyclic words $c(w)$ and $c(w')$ we test their equivalence by attempting to construct an isomorphism $\iota : c(w) \to c(w')$ such that $\iota(a) = a$ for all $a \in c(w)$. Clearly $|c(w)| = |c(w')|$ is a necessary condition for the existence of ι. If and only if ι exists, the cyclic words are considered equal, $c(w) = c(w')$. If and only if $c(w) = c(w')$, we have $w \approx_c w'$ by proposition 4.10. The set $c(w)$ may be visualized as the word w "made circular" as in figure 30.

The existence of ι may be established by testing the members of $c(w)$ for equality with the members of $c(w')$ pairwise in the following manner: Select from $c(w)$ an arbitrary member, a say. Check a for equivalence with all members of $c(w')$. Clearly, if and only if there exists a $b \in c(w')$ such that $a = b$, an ι exists. Since every word has length m and there are m words in $c(w')$, this comparison will take a time proportional to m^2. Thus it is possible to test the equivalence of two cyclic words of length m with complexity $O\left(m^2\right)$.

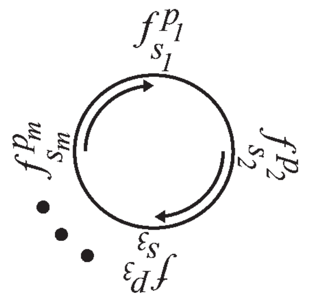

Fig. 30. The word w given in equation (59) bent into a circle. While the circularity removes the notions of beginning and end of a word, it preserves the directionality of it.

Rewriting Systems for Cyclic Words We shall call a TRS *cyclicly terminating*, *cyclicly confluent* and *cyclicly locally confluent* if it is respectively terminating, confluent and locally confluent under application to all cyclic words over the alphabet of the TRS. It is obvious from the above discussion that a cyclicly complete TRS solves the conjugacy problem. For this reason it is important to develop results about cyclic completeness along the lines of the results for linear words in order to obtain a conjugacy solution.

Termination in Cyclic Rewriting Systems We have seen that a TRS $\mathcal{R}$ terminates if and only if a reduction order exists [8]. In what follows, we shall assume that this reduction order is a total order; note that this is a stricter requirement than that of a reduction order. Suppose that the alphabet of $\mathcal{R}$ is $\mathcal{A} = \{f_i\}$ for $1 \leq i \leq p$. By assumption, p is finite. Consider the total order $<_{\mathcal{R}}$ defined by $f_i <_{\mathcal{R}} f_{i+1}$ for all i. This can be done without loss of generality as a mapping from $\mathcal{A}$ to itself can change the order. Recall that $\mathcal{R}$ terminates if and only if $r_i <_{\mathcal{R}} l_i$ for every rule $l_i \to r_i \in \mathcal{R}$.

We introduce an integer valued *weight* metric function $g(w)$ and an integer valued *length* metric function $L(w)$ on the set of words w written on the alphabet $\mathcal{A}$. The metrics satisfy

$$g(f_{a_1} f_{a_2} \cdots f_{a_m}) = g(f_{a_1}) + g(f_{a_2}) + \cdots + g(f_{a_m}) \tag{71}$$

$$L(f_{a_1} f_{a_2} \cdots f_{a_m}) = L(f_{a_1}) + L(f_{a_2}) + \cdots + L(f_{a_m}) \tag{72}$$

$$L(f_i) = 1 \tag{73}$$

$$g(f_i) < g(f_{i+1}) \tag{74}$$

We shall call a rule *length reducing* if $L(r_i) < L(l_i)$ and *weight reducing* if $g(r_i) < g(l_i)$. Any rule is a *c-obstruction* (for commutation-obstruction) if and only if it keeps constant both length and weight. That is, it is a rule which changes the position of the letters only.

A c-obstruction obstructs cyclic termination as there exist cyclic words which would give rise to an infinite rewriting chain due the changing of relative position of subwords by the c-obstruction. An example is the cyclic word $c(\alpha\beta\alpha\beta)$ under the TRS $\mathcal{R} = \{\alpha\beta \rightarrow \beta\alpha\}$. The rewriting chain will loop between the two states $c(\alpha\beta\alpha\beta)$ and $c(\beta\alpha\alpha\beta)$; the period of the loop may, in general, be arbitrarily large. Such looping may be dealt with in two ways. Firstly, one may compare each new cyclic word with the entirety of the rewrite chain so far enumerated. If equality is found, looping has been detected and one may stop. Secondly, one may determine if a subword of the current word commutes with the rest of the word. If this can be determined and such a subword is found, the subword may be extracted from the word and the two words should then be rewritten separately. The first method is computationally expensive and does not produce a unique normal form as we would have to consider the entire loop at the end of the rewrite chain as the identifying set of the word. The second method is not necessarily applicable but if it is, it will terminate in a set of subwords which uniquely identify the word. The advantage of the second method over the first is that the number of elements in the set has an upper bound.

The braid groups, as we have seen, have non-trivial center. In fact the generator of the center contains every generator. This fact makes it possible to rewrite any inverse generator in terms of inverses of the generator of the center multiplied by generators. As in the word problem case, we shall remove the generators of the center from the word and thus this replacement rewrites the entire braid word with which we shall work in terms of generators only. For the splitting of words to get rid of c-obstructions to fail we need to be in a situation in which we have a word abc such that $abc \approx cab$, $ab \approx ba$, $ac \not\approx ca$, $bc \not\approx cb$. This can only occur if there is cancellation between a and b in the word ab but this can not happen if there are no inverse generators. This proves that in groups in which inverse generators may be replaced by inverses of elements of the center and generators, this method of overcoming c-obstructions is valid.

We conjecture that a TRS $\mathcal{R}$ cyclicly terminates if and only if it terminates and contains no c-obstructions or contains c-obstructions that can be removed in the above way.

Confluence in Cyclic Rewriting Systems Newman's Lemma [62] extends easily to the cyclic case as we show below.

Lemma 4.11. *A cyclicly terminating TRS $\mathcal{R}$ is cyclicly confluent if and only if it is cyclicly locally confluent.*

Proof. The proof is similar to the standard proof (see [41]) and follows immediately from figure 31.

The Critical Pair Lemma states that a TRS is locally confluent if and only if it has no critical pairs. Recall that a critical pair arises from an overlap

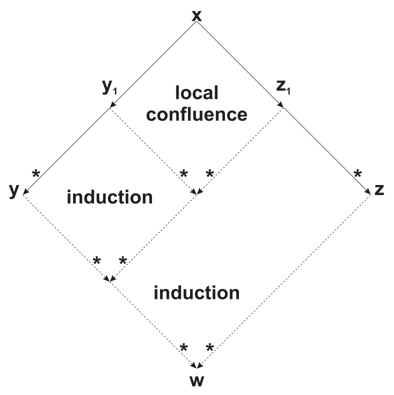

Fig. 31. The proof of Newman's Lemma (lemma 4.11) in diagrammatic form. We begin at the top with a local divergence which is rectifiable by assumption and thus by induction any global divergence is also rectifiable. It is because of this diagrammatic proof that Newman's Lemma is also known as the Diamond Lemma.

of two redexes in a word which gives rise to a local divergence of rewriting paths which do not meet again. Given a TRS $\mathcal{R} = \{(l_i, r_i)\}$, 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, $c(\rho r_i d) \leftrightarrow^{\star}_{\mathcal{R}} c(b\eta r_j)$ 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 prefered 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* whereas the usual rules of normal TRS's are *local*. We shall now prove the extention of the Critical Pair Lemma for the cyclic case.

Lemma 4.12. *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. This can easily be checked by going through all possible types of overlap.

It should be emphasized that the proof lemma 4.12 does not make any assumptions about the termination of $\mathcal{R}$. So we have a definite method for attempting to find a conjugacy problem solution in terms of rewriting. We shall use the braid groups to give an example of this completion process.

4.5 Markov's Theorem

Recall that Markov's theorem says

Theorem 4.13 (Markov). *Two braids $\alpha \in B_n$ and $\beta \in B_m$ have isotopic closures if and only if α can be transformed into β by a finite number of applications of conjugacy and stabilization moves.*

Corollary 4.14. *The closure of the braid $\alpha \in B_n$ is isotopic to the unlink of k components if and only if α can be transformed into the trivial braid in B_k by using conjugacy and stabilization moves.*

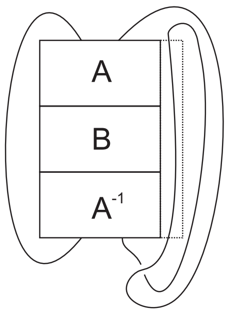

Fig. 32. Both conjugacy and stabilization are displayed here. We begin with braid B. Conjugation surrounds B with A and A^{-1} on opposite sides which clearly cancel due to the closure. Stabilization introduces a simple loop at the bottom right of the braid, adds a new string to the braid and thus increases the braid group index by one.

Conjugacy was discussed at length already. Stabilization is the move $\alpha \leftrightarrow \alpha\sigma_n$ with $\alpha \in B_n$ (see figure 32). While the conjugacy move is a move within a particular braid group, the stabilization move connects two adjacent braid groups. Therefore the question of detecting closed braid equivalence turns into a combinatorial question about the infinite family of braid groups.

Given a knot, we may produce an equivalent knot by taking any segment and twisting it about an axis in the projection plane by π while keeping the rest of the knot stationary. This procedure corresponds to the zeroth Reidemeister move and adds one crossing to the diagram. Any crossing of this type is called *nugatory*. If we represent a knot by a closed braid by virtue of Alexander's theorem, we may also add such nugatory crossings via a combinatorial move, called the *Markov* or *stabilization* move. Stabilizing a braid $\alpha \in B_n$ corresponds to the operation $\alpha \to \alpha\sigma_n^{\pm 1}$ or its inverse. Clearly stabilization increases or decreases the number of strings in the braid and so represents a move in the family of braid groups as opposed to the conjugacy and equivalence moves which are contained in a single braid group.

Markov stated in 1935 [55] that two closed braids are topologically equivalent if and only if they differ by stabilization and conjugacy moves (recall that conjugacy contains equivalence). This statement became known as Markov's theorem and was first proven in [14]. In its original form, Markov's theorem assumes that the closed braid is embedded in S^3 or R^3, this can however, be generalized to an arbitrary 3-manifold [51]. Markov's theorem transforms the link isotopy problem to a combinatorial question about braids. If two braids $\alpha \in B_n$ and $\beta \in B_m$ (with n and m possibly different) are related by stabilization and conjugacy, they are called *Markov equivalent* which is denoted by $\approx_M$. The decision problem of whether $\alpha \approx_M \beta$ is called the *Markov problem* or the *algebraic link problem.* It is possible to find a single move of which both stabilization and conjugacy are special cases and to formulate, in this way, Markov equivalence in terms of this so called L-move [52]. While this L-move is intuitive, it is not obvious whether the problem has been simplified by this reformulation.

The first question which arises is whether there exist non-conjugate Markov equivalent braid words in the same braid group, that is whether a solution to the conjugacy problem will solve the Markov problem. This is negatively resolved by showing that the two 4-braids $\alpha = \sigma_1^m \sigma_2^n \sigma_1^p$ and $\beta = \sigma_1^m \sigma_2^p \sigma_1^n$ with m, n, p different, odd and at least three in absolute value are not conjugate but Markov equivalent [61]. It might be thought that it should be possible to reduce the number of strings in a closed braid equivalent of the unknot to one. This is true as all equivalent closed braids can be reached from each other via Markov's theorem but the transition involves, in general, increasing the number of strings before they may be reduced to a single string. In other words, a greedy reduction of strings does not reach the minimum string number, also known as the braid index (not even for the unknot representatives) [58].

It is a practical observation that finding a series of moves to demonstrate the Markov equivalence of two closed braids is very difficult. The difficulty of finding such a sequence has lead Birman to believe that it may be simpler to solve Markov equivalence for two braids representing prime knots. While this may be true, it is not, in general, easy to decide whether a braid represents a prime knot. Schubert [67] proved that the factorization sequence of a composite knot is unique and has found an algorithm [68] which finds it. This algorithm, consequently, is able to decide whether a knot is prime. However, the execution of the algorithm rests on Hemion's algorithm since it must identify the prime factors of the knot, thus no longer necessitating a solution of the Markov problem since it already solves the link isotopy problem (albeit impractically so). This also shows that this method of deciding primality is not practical. Birman conjectures that a braid represents a prime knot if and only if it is not conjugate to a split braid.

Furthermore, if Birman's conjecture is true and we were to find an algorithm to decide whether a braid was conjugate to a split braid, we would have to solve the Markov problem for this restricted class of braids. If this could be done, we would have a solution to the Markov problem since every braid could be decomposed into its split components and pair-wise tested for non-split Markov equivalence. This would not only resolve isotopy but also give the unique prime knot factorization of the knots. Birman's conjecture is unproven and there exists no algorithm to test whether a braid is conjugate to a split braid. It is possible, however, to solve the Markov problem for certain quotient groups of the braid groups [23].

Since the word and conjugacy problems are contained in the Markov problem, solutions for these are desirable and have been given numerous times as mentioned before. The stabilization move represents the final hurdle before link isotopy is algorithmically decidable and thus it would be interesting to know when a braid $\alpha \in B_{n+1}$ is conjugate to a braid $\gamma\sigma_n^{\pm 1}$ where γ contains only the generators σ_i for $1 \leq i \leq n-1$, for then one could reduce α to γ using the Markov move. While this has been done [56], the algorithm depends on Garside's conjugacy algorithm [37] which has exponential complexity. Moreover, if two braids were reduced in this way to the minimum string number, they are not, in general, conjugate in this final braid group if they are Markov equivalent and thus this decision procedure does not solve the Markov problem either.

We have defined the exponent sum $exp(\alpha)$ of a braid α as the sum of the exponents of the Artin generators of α. It is obvious that the exponent sum is a conjugacy class invariant but not a Markov class invariant because of stabilization. Thus it is possible for two braids to be Markov equivalent and have different exponent sums. In getting from one braid to the other, the exponent sum must be made equal somewhere in the chain of moves; this can clearly only be accomplished using stabilization. Stabilization can increase or decrease the exponent sum depending whether we add σ_n or σ_n^{-1} or remove either of these. It also changes the number of strings. We may think that starting from a positive braid, we should be able to reach any Markov equivalent positive braid by going through a pure positive sequence of braids; that is, we may think that positive Markov equal braids are positively Markov equal. We note that this would only be possible if the difference in exponent sum between the two braids was precisely their difference in number of strings. We conjecture that positive Markov equal braids are not positively Markov equal.

Much work was done by Birman and Menasco on various properties of links which could be determined from their closed braid representatives (this work was published in the six-paper series [16], [17], [18], [19], [20] and [21]). They prove that there exists a complete numerical invariant for knots but find this invariant only for knots which are closed 3-braids. The invariant for closed 3-braids is described extensively and can be used to determine the braid index and whether the knot is split, composite, amphicheiral or invertible. They also define a new type of move on braids, the exchange move, and prove a Markov-like theorem for it. See [22] for a summary of this work.

4.6 The Minimal Word Problem

A well-known problem of combinatorial braid theory is the minimization problem: Given a braid $A \in B_n$ find a braid A_m such that $A \approx A_m$ and $L(A_m) \leq L(A^*)$ for any braid $A^* \approx A$ where $L(A)$ denotes the word length of the braid A.

In the Artin representation of B_n, the number of generators required to write down a braid word, its length, is equal to the crossing number of the topological braid. In practice, we find that by moving a few of the strings of the topological braid, its crossing number may be reduced, making the braid simpler. It would be especially useful to possess a general method to compute an equivalent braid of minimum crossing number. Apart from many applications, this problem is well-known in combinatorial braid theory and is of independent mathematical interest.

Given a braid $A \in B_n$ in the Artin generators, the question whether there exists an equivalent braid $A' \in B_n$ of shorter length has been shown to be NP-Complete by Paterson and Razborov [63]. Not only does this mean that this question is computationally equivalent to all other NP-Complete problems, it also means that (unless P = NP) any algorithm which answers the question would execute in exponential-time in n. Since Paterson and Razborov's result refers to the minimization problem for general n, we ask whether it is also an NP-complete question for particular n. This question is explicitly asked as open question 9.5.6 on page 209 of [36] and it seems to have been negatively answered in an unpublished preprint by Tatsuoka five years earlier but we were unable to obtain it [69].

In proving the NP-Completeness of the problem, Paterson and Razborov showed that the problem can be reduced to a known NP-Complete problem. This does not however provide a usable algorithm. For 3-braids, a linear complexity algorithm has been found [11] but no general algorithm for $n > 3$ exists. A minimization algorithm in the band-generator presentation of the braids groups has been found for $n = 3, 4$ but the length of the braid in this presentation is not equal to the crossing number [76] [47]. It is untypical of a group for which the word problem is solvable that no unique normal form of minimal length in some naturally arising presentation exists for the braid groups. A unique normal form of minimal length in certain natural presentations of free groups, HNN-extensions and free products exists, for example.

After a little experimentation, it is clear that a braid must, in general, be increased in length before it may be reduced to minimum length algebraically. We show that a certain readily obtained braid provides an upper bound for this necessary increase in length and prove several properties of this braid. We explicitly construct a set of words which must be searched for a certain property in order to obtain a minimal length representative of any braid. This constitutes an algorithm to solve the minimization problem. Since the set of words which must be searched is, in the worst case, exponential in size, the algorithm takes an exponential amount of time to complete.

Exercise 4.15. Find a braid which is non-minimal in length and which must be increased in length (by introducing pairs like $\sigma_i\sigma_i^{-1}$) before it can be shortened to minimal length. An example is the braid $\sigma_2\sigma_1^{-2}\sigma_2^{-2}\sigma_1\sigma_2^{-1}\sigma_1\sigma_2^2\sigma_1\sigma_2$.

Denote by A_m any braid which satisfies $A_m \approx A$ and $L(A_m) \leq L(A^*)$ for all braids $A^* \approx A$. We now prove a basic lemma which connects A_{max} and A_m. Recall that $A_{max} = \Delta_n^{-s(A)}A'$ where $s(A)$ is the number of inverse generators in A and A' is positive.

Theorem 4.16. *For any braid A, it is possible to obtain A_m from A_{max} by operations which monotonically decrease or keep constant the length of the braid.*

Proof. By construction $A_m \approx A_{max} \approx A$ and A_{max} and A are at least as long A_m. Exponent sum is an equivalence class invariant so that $s(A_m) \leq s(A)$. Replace each inverse generator in A_m with the braid given in proposition 4.1 and then use equation $\sigma_i\Delta_n = \Delta_n\sigma_{n-i}$ to bring all the fundamental braids to the front to obtain the braid

$$\begin{aligned} A_{m_{max}} &= \Delta_n^{-s(A_m)}A'_m && (75)\\ &\approx \Delta_n^{-s(A_m)}\Delta_n^{s(A_m)-s(A)}\Delta_n^{s(A)-s(A_m)}A'_m && (76)\\ &\approx \Delta_n^{-s(A)}\Delta_n^{s(A)-s(A_m)}A'_m && (77)\end{aligned}$$

But $A_{max} = \Delta_n^{-s(A)}A'$ and since the braid groups are left-cancelative [37], we have that

$$\Delta_n^{s(A)-s(A_m)}A'_m \approx A' \qquad (78)$$

with both words positive. Since positive words are positively equal [37], there exists a sequence of braids B_i for $0 \leq i \leq q$ with $B_0 = A'$, $B_q = \Delta_n^{s(A)-s(A_m)}A'_m$, B_j and B_{j+1} different by a single application of the braid group's defining relations and B_i positive for all i. Since exponent sum is an equivalence class invariant, $L(B_i) = L(A')$ for all i.

From A_{max} we may thus reach the form of $A_{m_{max}}$ in equation (77) keeping the length of the braid constant. From this form, we may reach A_m by operations which monotonically decrease or keep constant the length of the braid.

Thus there exists a sequence of braids W_i for $1 \leq i \leq p$ with $W_0 = A_{max}$, $W_p = A_m$, W_j and W_{j+1} different by a single application of the braid group's defining relations and $L(W_{j+1}) \leq L(W_j)$, which proves the lemma. □

Theorem 4.16 basically establishes that we may reach a minimum length representative from A_{max} by rearranging and cancelling generators only; it thus, in principle, removes the difficulty we pointed out in the introduction of occasionally having to increase the length before being able to decrease it to an absolute minimum.

We present an extension to the Cayley diagram construction which draws the diagram of any braid word (as opposed to positive braid words only). The diagram is a list of all those braid words which may be obtained from the given word by rearranging only.

Algorithm 4.17 *Input: A braid word A. Output: A list $D(A)$ of all braid words B which may be obtained from A by rearranging of generators only.*

1. Define the diagram of zeroth order as the set $D_0(A) = \{A\}$.
2. The set $D_i(A)$ is obtained from the set $D_{i-1}(A)$ by the following procedure:
   a) Fix attention on a particular member α of $D_{i-1}(A)$. We read α from left to right and decide at each position whether we may apply any of the moves in equations (79) to (82).

$$\sigma_i \sigma_j \leftrightarrow \sigma_j \sigma_i \text{ for } |i-j| > 1 \tag{79}$$

$$\sigma_i \sigma_{i+1} \sigma_i \leftrightarrow \sigma_{i+1} \sigma_i \sigma_{i+1} \tag{80}$$

$$\sigma_i \sigma_i^{-1} \leftrightarrow \sigma_i^{-1} \sigma_i \tag{81}$$

$$\sigma_i \sigma_i^{-1} \sigma_j \leftrightarrow \sigma_j \sigma_i \sigma_i^{-1} \tag{82}$$

   b) If we may, we apply it and store the resultant braid word β in $D_i(A)$ if and only if β is not already contained in $D_j(A)$ for $0 \leq j \leq i$.
   c) We continue to read across α until we have considered all braid words which may be reached from α by a single application of the moves in equations (79) to (82).
   d) Apply steps (a) through (c) for every braid in $D_{i-1}(A)$. If $D_i(A) = \emptyset$, then the algorithm is done.
3. The diagram $D(A)$ of A is the union of all the $D_i(A)$,

$$D(A) = D_0(A) \bigcup D_1(A) \bigcup \cdots \bigcup D_m(A) \tag{83}$$

We show the correctness and termination of this algorithm.

Lemma 4.18. *Algorithm 4.17 terminates for every A and succeeds in listing all braid words B which may be obtained from A by rearranging of generators only, that is using the braid group relations without introducing or removing any generators.*

Proof. $D_0(A)$ is, by definition, finite. It is obvious that for any braid word of finite length, the moves in equations (79) to (82) may be applied a finite number of times. Thus, by induction, every $D_i(A)$ is finite. The number of distinct braid words of a given finite length is finite and since the $D_i(A)$ are, by construction, non-overlapping, their union must be finite. Thus there exists an m such that $D_{m+k}(A) = \emptyset$ for every $k > 0$. Thus the algorithm terminates for every A.

The moves listed in equations (79) to (82) exhaust all possibilities allowed in the braid group under the stipulation that no generators must be removed from or introduced into the word. Thus each word which may be reached from A by rearrangement of generators will eventually be reached by algorithm 4.17 and so the algorithm succeeds in listing all the required braid words. $\square$

Theorem 4.16 gives the following corollary.

Corollary 4.19. *$D(A_{max})$ contains a braid of the form EA_m for $E \approx e$, the identity in B_n.*

Proof. By construction $D(A_{max})$ contains all braid words equivalent to A_{max} by rearranging only. By lemma 4.16, A_m can be obtained by a sequence of operations which keeps the length constant or decreases it. Each operation which decreases the length does so by eliminating a sub-word like $e_i = \sigma_i \sigma_i^{-1} \approx \sigma_i^{-1}\sigma_i$.

Since for all i $e_i \approx e$, the identity in B_n, we have

$$e_i \sigma_j^{\pm 1} \approx \sigma_j^{\pm 1} e_i, \qquad e_i e_j \approx e_j e_i \tag{84}$$

for any i and j.

Let us now agree to construct the aforementioned sequence of words without eliminating the sub-words e_i but using equation (84) to bring them all to the left of the word. At the end, we will obtain a word of the form $A^* = EA_m$ where $E \approx e$ is a braid consisting of all these sub-words e_i. The most general form of E is

$$E = e_1^{q_1} e_2^{q_2} \cdots e_{n-1}^{q_{n-1}} \tag{85}$$

with $q_i \geq 0$ for all i. So if we could extract E from A_{max}, we would, in the process, obtain A_m. Since the form EA_m is obtained by rearrangements only, $L(E) \leq L(A^*) = L(A_{max})$. This indicates that $\sum_{i=1}^{n-1} 2q_i \leq L(A_{max})$. $\square$

Given a braid A, we thus find A_m by constructing the diagram $D(A_{max})$ and selecting the word with the largest number of cancellation pairs such as $\sigma_i \sigma_i^{-1}$. Clearly there will be more than one braid word for the same number of cancellation pairs. We may agree to choose the least braid word lexicographically for definiteness. It is obvious from the construction that this will be a *unique* form of minimal length for the braid A. We thus have an algorithm to find A_m for any A. It is regrettable that the diagram $D(A_{max})$ is, by construction, very large. Two questions are left to ask: Can we make the result stronger and how large is a typical diagram?

In theorem 4.16 we achieved an upper bound for the necessary increase in length of a braid before it may be reduced to a minimum length. One would like to simplify the result somewhat but we shall show in this section that the two straightforward attempts to simplify or strengthen theorem 4.16 are doomed to failure. First we show that we may not, in general, shorten A_{max} to the Garside normal form.

Lemma 4.20. *It is not, in general, possible to obtain A_m from $G(A)$, the Garside normal form of A, by operations which monotonically decrease or keep constant the length of the braid.*

One may think that it would be sufficient to list the diagram of the negative and positive sub-braids of A_{max} and search for a maximal length sub-braid which is common to the end of the first and the beginning of the second diagram but this is not true as the following lemma shows.

Lemma 4.21. *There does not exist an A_m in the form $A_1 A_2$ with A_1 negative and A_2 positive for every A.*

Let a be a n-braid of length L with diagram $D(a)$. Consider the braid $a' = a\sigma_i\sigma_i^{-1}$ for some $1 \leq i < n$. We are concerned with the size of $D(a')$ in terms of the size of $D(a)$. For each member of $D(a)$, the cancellation pair $\sigma_i\sigma_i^{-1}$ may appear in any place in both possible orders ($\sigma_i\sigma_i^{-1}$ and $\sigma_i^{-1}\sigma_i$), so in $2(L+1)$ positions. There may be further moves possible by use of the braid group relations but the number of these are clearly bounded by a function linear in L. So the diagram of a word will increase in size by a factor linear in its length for each possible cancellation pair. Given a random positive n-braid a of length L, how many members will $D(a)$ have, on average? We conjecture that:

Conjecture 4.22. For any braid $a \in B_n$ of length L, we have that $|D(a)| \leq |D(\Delta_n^p)|$ with $p = \lceil 2L/(n(n-1)) \rceil$.

Conjecture 4.22 would provide an upper bound for the size of the diagram of any word in terms of the diagrams of the diagrams of Δ_n^p which topologically are a series of p half-twists of the braid strings about the vertical axis. In extensive computer simulations, the conjecture was checked and seems to hold. What it seems to indicate is that the half-twist has the most topological freedom for its length and number of strings under the constraint that the crossing number must be kept constant. This is quite intuitive, yet the conjecture seems to be difficult to prove.

We have investigated the diagrams of several Δ_n^p for their size and for the distribution of braids over the sub-diagrams at each stage of the construction in algorithm 4.17. In table 1 we list the size of the diagram and maximal sub-diagram index for p half-twists on n strings.

Table 1. The Size of Diagrams of Fundamental Words

n	p	$\lvert D\left(\Delta_n^p\right)\rvert$	max. i
3	1	2	1
3	2	8	2
3	3	38	5
3	4	196	8
3	5	1062	13
3	6	5948	18
3	7	34120	25
4	1	16	7
4	2	1654	15
5	1	768	25

We conclude that the diagram of a typical braid word grows exponentially with its length and braid index and thus our method of finding the minimal length braid word equivalent to a given braid has exponential complexity. This is not surprising as the problem is NP-Complete. We shall give a heuristic algorithm and other methods later. The properties of the braid groups that made the above solution possible are: (i) It is possible to write all inverse generators as products of the generator of the center and a positive word, (ii) the defining relations relate positive words only and (iii) the braid groups are right and left-cancellative. It is likely that any group which has these properties, has an analog of the Garside normal form and has a solution to the minimum word problem similar to the one above.

Solving the problem exactly is an expensive endeavor and so we ask for approximate methods. It turns out that magnetic relaxation is an important application of this problem and gives rise to good methods to solve it. We shall delay the discussion of these to section **??**. It is possible to solve the problem heuristically using a purely algebraic algorithm which we now present.

Recall that the braid group B_n is defined by

$$B_n = \langle\ \{\sigma_i\} :\ 1 \leq i < n; \tag{86}$$

$$\sigma_i \sigma_j = \sigma_j \sigma_i \ |i-j| > 1; \sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1} \rangle\ . \tag{87}$$

An n-braid A of c crossings is a word in B_n of word-length c, so the general form of A is

$$A = \sigma_{a_1}^{\epsilon_1} \sigma_{a_2}^{\epsilon_2} \cdots \sigma_{a_c}^{\epsilon_c} \qquad \epsilon_k = \pm 1,\ 1 \leq a_k < n,\ \forall k : 1 \leq k \leq c. \tag{88}$$

Consider an n-braid A of the form given in equation 88. Suppose we wish to find the n-braid A_m equivalent to A such that the length $L(A_m)$ of A_m is minimal over the equivalence class of A. It has been shown [63] that this question is NP-complete and hence computationally difficult (if P $\neq$ NP, it is intractable). The following presents a heuristic algorithm for getting close to A_m. We begin with the leftmost generator of A and attempt to move it to the right using both braid group operations. If we can cancel it along the way, we do and if we can not, we move it back to where it started. In this way, we proceed to move all the generators as far to the right as possible. Then we begin at the end and move each generator as far to the left as possible in the same manner. This algorithm will always produce an equivalent braid A' such that $L(A') \leq L(A)$. We consider $L(A)$ generators and move them $O(L(A))$ moves to the right and left. Thus this algorithm takes $O\left(L(A)^2\right)$ time and constant memory. In fact we move a particular generator at most $L(A)$ generators and this is only for the case when all the other generators commute with it, thus the average case complexity is likely to be close to linear in $L(A)$.

This algorithm will not produce a minimum length representative in all cases because it can not unravel complex crossings. To get to the minimum length would require more subtle transformations than just movements to the right or left, which topologically correspond to pulling the strings apart from underneath the crossing. However, as computer experiments show, it does do quite well.

Let us calculate an upper bound to the reduction ratio obtained by this method as a function of n and c. To calculate these, consider the likelihood that a particular generator will be followed by its inverse, which is just $Q_0 = 1/2(n-1)$. The probability Q_j that a generator and its inverse are separated by j generators through which either can be moved is the corresponding probability for $j = 1$ to the power j. We require the number of braids of length 1 which may be generated so as not to contain the generator interfering with the movement of generator σ_i. If $i = 1$ or $n-1$, this is $2(n-3)$ and $2(n-4)$ otherwise. Thus

$$Q_j = \left[\frac{2(n-4)\left(\frac{2(n-1)-2}{2(n-1)}\right) + 2(n-3)\left(\frac{2}{2(n-1)}\right)}{2(n-1)}\right]^j Q_0 \tag{89}$$

$$= \left[\frac{n^2-5n+5}{(n-1)^2}\right]^j Q_0 \tag{90}$$

The final factor of Q_0 is present because the generator after the sequence of j generators is required to be inverse of the original generator, an event with probability Q_0. To get the total probability Q of being able to cancel a generator σ_i with its inverse by simple exchange movements over the length $j = 0, 1, \cdots$, we must sum these probabilities in order weighted by the probability that their predecessors did not happen. Thus

$$Q = Q_0 + (1-Q_0)Q_1 + \cdots + \prod_{k=0}^{j-1}(1-Q_k)Q_j + \ldots \tag{91}$$

Note that since the exchange move is not allowed for $n = 3$, $Q = Q_0$ for $n = 3$. The reduction ratio R which occurs as a consequence of this probability is

$R = 1 - 2Q$ since each time that the event happens two generators may be canceled. Note that in this calculation we have considered the probability that a generator can be moved next to its inverse in the word using only the far commutation relation that $\sigma_i\sigma_j = \sigma_j\sigma_i$ for $|i-j| > 1$ in a long braid. The heuristic algorithm however uses both braid group moves to attempt to move generators next to their inverses. Thus R is an upper bound for the reduction ratio achieved by the heuristic algorithm as the braid becomes long.

In §6 we present the results of the algebraic reduction of a large number of braids but a few comments about the efficiency of the algorithm are in order. The only exact algorithm to minimize braid is valid only for $n \leq 3$ [11] and by comparing this heuristic to this exact algorithm, we find that the heuristic finds a braid the length of which is within five percent of the length found by the exact algorithm and that it reaches the actual minimum in 0.005 of all cases. This shows that the heuristic is quite effective for $n = 3$ (note that reduction for $n = 1, 2$ is trivial since B_1, B_2 are free groups).

5 OpenProblems

The problems that follow are, to the best of my knowledge, unsolved at the time of publication. The selection is personal and far from complete. However, each problem is quite significant in that its solution will have an impact on research. I believe that even partial results for most of these problems would be worth a Ph.D. The problems are presented in no particular order.

1. Is the Burau representation faithful for $n = 4$? Seems hard to answer but the significance is somewhat debatable since it is unfaithful for $n > 4$.
2. Does the Jones polynomial identify the unknot? In other words, is there a knot (distinct from the unknot) that has the same Jones polynomial as the unknot? (Warning: Searches for a counterexample have gotten quite far in the knot tables, so if there is a counterexample it is not simple. At present, it seems likely that there is no counterexample.)
3. Construct a polynomial-time algorithm for calculating the Jones and related polynomials or show that this cannot be done (assuming that $P \neq NP$ a proof of NP-completeness would do).
4. Construct an inherently algebraic algorithm for solving the Markov problem. (Note: The problem is definitely solvable via Hemions algorithm but this is very complicated, we are looking for a simple algorithm even if it is in exponential-time.)
5. Prove that the Markov problem is NP-complete, intractable or find a polynomial-time algorithm for it. (Note: Depending on how this is done and the answer, this is probably a Fields Medal problem but that should not discourage you from trying it.)
6. Construct a provably secure cryptosystem on the basis of the braid or similar groups. (Note: With present systems the question of security is not quite settled.)

7. Create a general theory of 2-tangle equations and how to solve them. (Note: The solvability needs to be practical as these equations actually come up in biology.) As an encore, do it for n-tangle equations.
8. Create a theory of physical braids with aims to study then tension properties of various knots independent of the rope on which they are tied.
9. Construct further invariants of knots and braids, preferably easily calculable and strong (not too many distinct knots having the same value of your invariant).
10. Construct a complete invariant of knots that can actually be compared computationally. (Note: The peripheral group system is complete but it cannot be compared to other systems because groups cannot in general be distinguished from another.)
11. Build a software system that makes it easy to input/output knots and braids, compute many invariants, identify the knot in a table, construct a knot table and play with new ideas largely independent of complex programming.

References

1. Adams, C., Hildebrand, M., Weeks, J. (1991): Hyperbolic Invariants of Knots and Links. Trans. Amer. Math. Soc., **326**, 1–56.
2. Adian, S. I. (1957): The unsolvability of certain algorithmic problems in the theory of groups. Trudy Moskov. Mat. Obsc., **6**, 231–298.
3. Adian, S. I. (1957): Finitely Presented Group and Algorithms. Dokl. Akad. Nauk SSSR, **117**, 9–12.
4. Alexander, J. W. (1923): A lemma on systems of knotted curves. Proc. Nat. Acad. Sci. USA, **9**, 93–95.
5. Appel, K. I. and Schupp, P. E. (1972): The Conjugacy Problem for the Group of any Tame Alternating Knot is Solvable. Proc. Amer. Math. Soc., **33**, 329–336.
6. Artin, E. (1925): Theorie der Zöpfe. (in German) Abh. Math. Sem. Univ. Hamburg, **4**, 47–72.
7. Artin, E. (1947): Theory of braids. Ann. Math., **48**, 101–126.
8. Baader, F., Nipkow, T. (1998): Term Rewriting and All That. (Cambridge University Press, Cambridge).
9. Bangert, P. D., Berger, M. A., Prandi, R. (2002): In Search of Minimal Random Braid Configurations. J. Phys. A., **35**, 43–59.
10. Berger, M. A. (1993): Energy-crossing number relations for braided magnetic fields. Phys. Rev. Lett., **70**, 705–708.
11. Berger, M. A. (1994): Minimum crossing numbers for 3-braids. J. Phys. A, **27**, 6205–6213.
12. Berger, M.A. (2000): Hamiltonian dynamics generated by Vassiliev invariants. J. Phys. A., **34**, 1363–1374.
13. Bigelow, S. (1999): The Burau representation is not faithful for $n = 5$. Geometry and Topology, **3**, 397–404.
14. Birman, J. S. (1974): Braids, Links and Mapping Class Groups. Ann. of Math. Studies 82 (Princeton Univ. Press, Princeton).

15. Birman, J. S., Ko, K. H., Lee, S. J. (1998): A New Approach to the Word and Conjugacy Problems in the Braid Groups. Ad. Math., **139**, 322–353.
16. Birman, J. S., Menasco, W. (1992): Studying Links Via Closed Braids I: A Finiteness Theorem. Pacific J. Math., **154**, 17–36.
17. Birman, J. S., Menasco, W. (1991): Studying Links Via Closed Braids II: On a Theorem of Bennequin. Topology and Its Applications, **40**, 71–82.
18. Birman, J. S., Menasco, W. (1993): Studying Links Via Closed Braids III: Classifying Links which are Closed 3-braids. Pacific J. Math., **161**, 25–113.
19. Birman, J. S., Menasco, W. (1990): Studying Links Via Closed Braids IV: Closed Braid Representatives of Split and Composite Links. Invent. Math., **102**, 115–139.
20. Birman, J. S., Menasco, W. (1992): Studying Links Via Closed Braids V: Closed Braid Representatives of the Unlink. Trans. AMS, **329**, 585–606.
21. Birman, J. S., Menasco, W. (1992): Studying Links Via Closed Braids VI: A Non-Finiteness Theorem. Pacific J. Math., **156**, 265–285.
22. Birman, J. S., Menasco, W. (1992): A calculus on links in the 3-sphere. in Kawauchi, A. *Knots 90* (Walter de Gruyter, Berlin), 625–631.
23. Birman, J. S., Wajnryb, B. (1986): Markov Classes in Certain Finite Quotients of Artin's Braid Group. Israel J. Math., **56**, 160–178.
24. Birkhoff, G. (1935): On the structure of abstract algebras. Proc. Camb. Phil. Soc., **31**, 433–454.
25. Bohnenblust, F. (1947): The Algebraical Braid Group. Ann. Math., **46**, 127–136
26. Boyland, P. L., Aref, H., Stremler, M. A. (2000): Topological fluid mechanics of stirring. J. Fluid Mech., **403**, 277–304.
27. Buchberger, B. (1987): History and Basic Features of the Critical-Pair/Completion Procedure. J. Sym. Comp. **3**, 3–38. Printed in *Rewriting Techniques and Applications* ed. by Jouannaud, J.-P. (Academic Press, London), 3–38.
28. Chan, T. (2000): HOMFLY polynomials of some generalized Hopf links. J. Knot Th. Rami., **9**, 865–883.
29. Cohen, D.E. (1987): Computability and Logic. (Ellis Horwood, Chichester).
30. Conway, J.H. (1970): An Enumeration of Knot and Links, and Some of their Algebraic Properties. in Leech, J. (1970): Computational Problems in Abstract Algebra. (Pergamon, Oxford)
31. Coxeter, H.S.M., Moser, W.O.J. (1957): Generators and Relations for Discrete Groups. (Springer, Berlin)
32. Dershowitz, N. (1979): A note on simplification orderings. Inform. Proc. Let., **9**, 212–215.
33. Dershowitz, N. (1981): Termination of linear rewriting systems. in *Automata, Languages and Programming* ed. by Even, S. and Kariv, O., Lecture Notes in Computer Science Volume 115 (Springer, Heidelberg), 448–458.
34. Dershowitz, N. (1987): Termination of Rewriting. J. Sym. Comp., **3**, 69–116. Printed in *Rewriting Techniques and Applications* ed. by Jouannaud, J.-P. (Academic Press, London), 69–116.
35. Dowker, C. H. and Thistlethwaite, M. B. (1983): Classification of Knot Projections. Top. Appl., **16**, 19–31.
36. Epstein, D.B.A., Cannon, J.W., Holt, D.F., Levy, S.V.F., Paterson, M.S., Thurston, W.P. (1992): Word Processing in Groups. (Jones and Bartlett, Boston).

37. Garside, F.A. (1969): The braid group and other groups. Quart. J. Math. Oxford, **20**, 235–254.
38. Gilbert, N.D., Porter, T. (1994): Knots and Surfaces. (Oxford University Press, Oxford).
39. Hemion, G. (1992): The Classification of Knots and 3-dimensional Spaces. (Oxford Univ. Press, Oxford).
40. Hempel, J. (1976): 3-manifolds. Ann. of Math. Studies Volume 86 (Princeton Uni. Press, Princeton).
41. Huet, G. (1980): Confluent reductions: Abstract properties and applications to term rewriting systems. J. Assoc. Comput. Mach., **27**, 797–821.
42. Huet, G. (1981): A Complete Proof of the Knuth-Bendix Completion Algorithm. J. Comp. Syst. Sci., **23**, 11–21.
43. Huet, G., Lankford, D.S. (1978): On the uniform halting problem for term rewriting systems. Rapport laboria 283, Institut de Recherche en Informatique et en Automatique, Le Chesnay, France.
44. Jacquemard, A. (1990): About the effective classification of conjugacy classes of braids. J. Pure Appl. Al., **63**, 161–169.
45. Johnson, D.L. (1980): Topics in the Theory of Group Presentations. Lon. Math. Soc. Lec. Notes Vol. 42 (Cambridge Uni. Press, Cambridge).
46. Jouannoud, J.P., Kirchner, H. (1986): Completion of a set of rules modulo a set of equations, SIAM. J. Comp., **15**, 1155–1194.
47. Kang, E. S. *et. al.* (1997): Band-generator presentation for the 4-braid group. Top. Appl., **78**, 39–60.
48. Kauffman, L. (1993): Knots and Physics. Series on Knots and Everything Vol. 1. World Scientific, Singapore.
49. Kawauchi, A. (1996): A Survey of Knot Theory. (Birkhäuser Verlag, Basel)
50. Knuth, D.E., 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.
51. Lambropoulou, S.S.F. (1993): A Study of Braid in 3-manifolds. unpublished PhD thesis (Univ. of Warwick).
52. Lambropoulou, S.S.F., Rourke, C.P. (1997): Markov's Theorem in 3-manifolds. Top. Appl., **78**, 95–22.
53. The CiME system is available from `http://cime.lri.fr`.
54. Magnus, W., Peluso, A. (1967): On Knot Groups. Comm. Pure Appl. Math., **20**, 749–770.
55. Markov, A.A. (1935): Über die freie Äquivalenz geschlossener Zöpfe (in German), Recueil Mathematique Moscou, **1**, 73–78. [Mat. Sb. **43** 1936.]
56. McCool, J. (1980): On Reducible Braids. in *Word Problems II* ed. by Adian, S. I., Boone, W. W. and Higman, G. (North-Holland, Amsterdam), 261–295.
57. 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.
58. Morton, H.R. (1983): An irreducible 4-string braid with unknotted closure. Math. Proc. Camb. Phil. Soc., **93**, 259–261.
59. Morton, H.R. (1986): Threading Knot Diagrams. Math. Proc. Camb. Phil. Soc., **99**, 247–260.

60. Murasugi, K. (1996): Knot Theory And Its Applications. (Birkhäuser, Boston).
61. Murasugi, K., Thomas, R.S.D. (1972): Isotopic closed nonconjugate braids. Proc. Am. Math. Soc., **33**, 137–139.
62. Newman, M.H.A. (1942): On theories with a combinatorial definition of 'equivalence'. Ann. Math., **43**, 223–243.
63. Paterson, M.S., Razborov, A.A. (1991): The set of minimal braids is co-NP-complete. J. Algorithms, **12**, 393–408.
64. Rabin, M.O. (1958): Recursive Unsolvability of Group Theoretic Problems. Ann. Math., **67**, 172–194.
65. Reidemeister, K. (1983): Knot Theory. BCS Associates, Moscow, Idaho. Originally published as Reidemeister, K. (1932): Knotentheorie. Springer, Berlin
66. Ricca, R.L. (1998): Applications of Knot Theory in Fluid Mechanics. in Jones, V.F.R. *et. al.*, ed. by *Knot Theory* Banach Center Pub. Vol. 42 (Inst. of Math., Polish Acad. Sci., Warszawa), 321–346.
67. Schubert, H. (1948): Die Eindeutige Zerlegbarkeit eines Knotens in Primknoten. (in German), Sitz. Heidelberger Akad. Wiss., math.-nat. Kl., 55–104.
68. Schubert, H. (1961): Bestimmung der Primfaktorzerlegung von Verkettungen. (in German), Math. Zeitschr., **76**, 116–148.
69. Tatsuoka, K. (1987): Geodesics in the braid group. preprint, Dept. of Mathematics, University of Texas at Austin.
70. Tourlakis, G.J. (1984): Computability. (Reston, Reston).
71. Turing, A.M. (1937): On Computable Numbers, with an Application to the Entscheidungsproblem. Proc. London Math. Soc. Ser. 2, **42**, 230–265.
72. Vogel, P. (1990): Representation of links by braids: A new algorithm. Comment. Math. Helvetici, **65**, 104–113.
73. Waldhausen, F. (1968): On Irreducible 3-manifolds which are Sufficiently Large. Ann. Math., **87**, 56–88.
74. Waldhausen, F. (1968): The Word Problem in Fundamental Groups of Sufficiently Large Irreducible 3-manifolds. Ann. Math., **88**, 272–280.
75. Williams, R.F. (1988): The Braid Index of an Algebraic Link. In: Birman, J. S., Libgober, A. (ed) Braids. Amer. Math. Soc., Providence.
76. Xu, P.J. (1992): The genus of closed 3-braids. J. Knot Theory Ramifications, **1**, 303–326.
77. Yamada, S. (1987): The minimal number of Seifert circles equals the braid index of a link. Invent. Math., **89**, 347–356.
78. Yoder, M.A. (1995): String Rewriting Applied to Problems in the Braid Groups. unpublished Ph.D. thesis (Uni. South Florida).