Knot Notation and Braiding[‡]

Patrick D. Bangert

May 3, 2004

Abstract

We use tangles in constructing a new notation for knots based on Conway's knot notation. The advantages of and basic manipulation algorithms for the new notation are given. This new notation allows the construction of a closed braid and closed plait representative for any given knot. The necessary increase in the number of crossings due to this conversion is much smaller than in previously known methods. This algorithm runs in $O(n)$ time whereas all previously known algorithms run in $O(n^2)$ where n is the number of crossings in the knot diagram. With the given methods, this notation is useful in computer manipulation of knots, particularly tabulating, enumerating and computing invariants of knots.

1 Introduction

Knot theory has gained tremendous momentum from proofs that certain mathematical objects are ambient isotopy invariants of knots. Such proofs and general statements about knots form a large part of knot theory but in applications of knot theory, actual computation of these objects (for example the Jones polynomial) is often necessary. Therefore, it is important to have a practical method of computation for such invariants. Some invariants can only be calculated by algorithms whose computation time increases exponentially with crossing number, thus rendering them practically useless for all but small knots. There exist only a few invariants which may be calculated easily and quickly for all knots.

Because it is so laborious to compute many interesting properties of a particular knot, the use of computers is essential. However, if a computer is to be used, the search for an efficient algorithm becomes important. The pivot of all algorithms is the form of the input. For many physics calculations, for example, the choice of coordinate system often allows far greater simplification of the calculations than a change in computational procedure. Therefore, while the algorithm is important, a good notation for knots is paramount. Currently

*School of Engineering and Science, International University Bremen, P.O. Box 750 561, 28725 Bremen, Germany; http://www.knot-theory.org; p.bangert@iu-bremen.de.

†Many thanks to Mitchell Berger for helpful discussions and support. I also thank the anonymous referee for a number of helpful criticisms and suggestions.

there are two different systems of "knotation" (the term was coined by John Conway in a popular lecture with this title) that are widely used: Conway's [5] and Dowker and Thistlethwaite's [6].

The Dowker-Thistlethwaite code for a knot is a series of numbers which indicates the order in which crossings are encountered when the knot is traversed in the direction of its orientation. This code is very compact in that only a few numbers are required to name large knots. Implemented algorithms to calculate most invariants from this code exist. The main application of this code is in the computer-assisted tabulation of knots [15]. Conway's knotation relies on setting up templates for knots and inserting standard knot pieces called tangles into the vertices of the template. This knotation is quite intuitive since the geometrical aspects of the knot projection can be immediately visualized.

In this paper, we will introduce a new knotation based on Conway's. We prove that all knots may be represented by it, give an algorithm to place a given knot into this notation and present a traversal algorithm which will calculate certain features of the knot. An algorithm is then given to obtain a plait and a braid, the closures of which are ambient isotopic to any given knot in the new notation.

2 Tangles

2.1 Definition and Partition

Consider the 3-ball B^3, choose $2n$ points on its surface, which is the 2-sphere S^2, and call the set of these points P. Attach n polygonal curves to the $2n$ points such that: (i) each curve intersects S^2 in exactly 2 points in P, which are its endpoints, (ii) exactly one curve may begin or end at any one point in P and (iii) no curve may intersect another. If the set of these curves is T, then we will call the set $\left(B^3, T\right)$ an n-tangle. In particular, we will focus on 2-tangles and so whenever we skip the n, it will be understood that we mean $n = 2$. Note that the requirement that the curves be polygonal excludes any wild tangles, where wild is to be understood in the usual knot theory sense. Two tangles are called *equal* if they are isotopic without moving the points in P.

A tangle can be visualized readily by choosing the four points (named according to the cardinal points of the compass)

$$NE = \left(0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right), \qquad NW = \left(0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right) \tag{1}$$

$$SE = \left(0, \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right), \qquad SW = \left(0, -\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}}\right) \tag{2}$$

on the unit sphere, which will be our canonical B^3, see figure 1. Even though tangles are, by definition, three dimensional objects, we will work with their diagrams in the two dimensional plane as if the diagram is the tangle. The fact that a projection in which there are at worst double points always exists for a

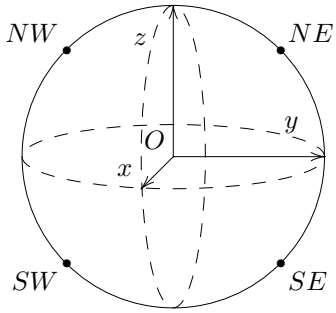

Figure 1: The 3-ball and the four points on its surface which form the endpoints of the two polygonal curves necessary to define a tangle.

tangle follows from the corresponding theorem about knots (we understand the term "knot" as inclusive of links with an arbitrary number of components).

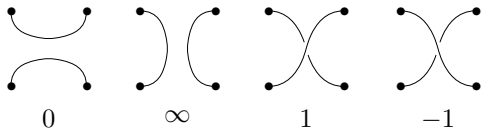

Figure 2: The elementary tangles.

We shall find it convenient to partition the set of all possible tangles into a few categories: elementary, integral, fractional and rational. The simplest are the *elementary tangles*, of which there are four. These are best introduced by displaying them in figure 2. Note that we have not drawn B^3, it should however be understood to be present. The reason for naming them as they have been will become apparent later on. Note that the literature disagrees on which of the two tangles ± 1 is to have the minus sign. This is a matter of convention and has no serious consequences (we follow the convention introduced by Conway).

The other types of tangles can be most readily defined in terms of combining the elementary ones in some way. To do this, we shall define two ways of adding tangles. Following Conway, we denote a general tangle by an "L" shaped symbol within the three ball and we also sketch the ends of the two curves by which tangles may be attached to each other. In this way, we define the horizontal sum $+$ and the vertical sum $\oplus$ in figure 3.

In what follows, we shall use a superscript to denote of which type a particular tangle t is; for example an elementary tangle t would be denoted by $t^{(e)}$.

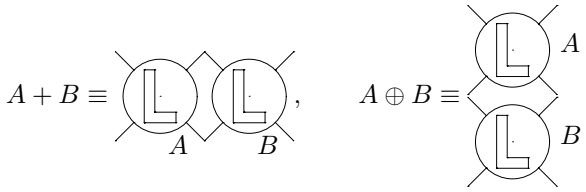

Figure 3: Tangle addition.

An *integral tangle* $t^{(i)}$ and a *fractional tangle* $t^{(f)}$ will be defined in terms of the elementary tangles ± 1 by

$$t^{(i)} = \underbrace{1+1+\cdots+1}_{t \text{ factors}} \tag{3}$$

$$t^{(f)} = \underbrace{1\oplus 1\oplus\cdots\oplus 1}_{t \text{ factors}} \tag{4}$$

The negative versions are, of course, the sums of -1 tangles instead of 1 tangles. A *rational tangle* $t^{(r)}$ can then be defined in terms of a sum of integral and fractional tangles. The definition of the sum differs if the number of tangles j in the sum is even or odd, this is because the definition requires an alternate sum between integral and fractional tangles (and the two methods of addition) which always *ends* in an integral tangle being added horizontally. This is because the set of rational tangles may be classified if this restriction is imposed; the classification scheme is outlined in the next section. The integral tangles, including the last, may be zero and the fractional tangles may be infinite. If any component tangles, except the last one, are 0 or ∞ though, they may be removed from the sum and the terms immediately preceding and following the removed term may be added together to shorten the sum, while preserving isotopy.

$$t^{(r)} = \begin{cases} a^{(i)} \oplus b^{(f)} + c^{(i)} \oplus d^{(f)} + \cdots + z^{(i)} & j \text{ odd} \\ a^{(f)} + b^{(i)} \oplus c^{(f)} + d^{(i)} \oplus \cdots + z^{(i)} & j \text{ even} \end{cases} \tag{5}$$

Note that the set of elementary tangles is a subset of both the integral and fractional tangle sets which are subsets of the rational tangle set.

2.2 Classification of Tangles

We may denote a rational tangle by giving its integral and fractional factors in order. Thus a sequence of integers $t^{(r)} = (a_1, a_2, \ldots, a_i)$ defines any rational tangle. Note again that the identity of the tangle factors is decided by requiring the last in the sequence to be integral. Given a rational tangle $t^{(r)} = (a_1, a_2, \cdots, a_i)$, we may associate with it an extended rational number $E(t^{(r)}) = \alpha/\beta$, where α

and β are integers including zero. We say an extended rational number because this allows for $1/0 = \infty$, the inclusion of which extends the rational numbers. We calculate $E\left(t^{(r)}\right)$ by the continued fraction

$$E\left(t^{(r)}\right) = a_i + \cfrac{1}{a_{i-1} + \cfrac{1}{a_{i-2} \cdots + \cfrac{1}{a_1}}} \tag{6}$$

Conway [5] was able to deduce that two rational tangles are isotopic if and only if the associated extended rational numbers were equal; this is called Conway's Basic Theorem. The first published proof may be found in [4] but a more intuitive proof was given by Goldman [8]. Thus Conway's Basic Theorem classifies rational tangles in a simple algorithmic manner.

In particular, the fractions associated with the elementary tangles are their numerical names: 0, ± 1 and ∞. The fraction for an integral tangle $t^{(i)}$ is $t^{(i)}$ and for a fractional tangle $t^{(f)}$ is $1/t^{(f)}$. It is clear now why these tangles were named as they were.

By equation 6, it is easy to calculate the fraction associated with a given rational tangle. Given a fraction, it is also possible to decompose it into appropriate factors, thereby constructing the rational tangle associated with it. This may be accomplished using Euclid's algorithm. This concludes our review of previous work on tangles and the rest of the paper is new work.

3 Knot Notation

Tangles were invented in an effort classify knots (they may be used to classify two-bridge knots via the correspondence with the extended rational numbers [11]) and so we must have a method to combine tangles to make knots. Conway [5] showed that any knot may be obtained by substituting several rational tangles into the vertices of basic polyhedra. A *polyhedron*, in the sense of Conway, is an edge-connected 4-valent planar map and it is *basic* if, in addition, no region (including the infinite region) has just two vertices. Conway constructs the 8 different basic polyhedra necessary to denote all prime knots up to and including 11 crossings [5].

Conway was able to deduce certain symmetries and functional relationships in this notation that lead him to discover what is now known as the Alexander-Conway polynomial. If one wants to give the construction of a particular knot, all one has to do is to give the polyhedron and specify the tangle fractions to be substituted into the vertices. The beauty of using the basic polyhedra is that small knots may be named quite efficiently. Moreover, by Conway's research, a complete list of knots in this notation is available up to and including 11 crossings. However, the notation begins to get cumbersome for larger knots. First, it is not easy to see how a knot is to be fitted into the basic polyhedra and there is no method given to determine what the smallest polyhedron is that a given knot can be fitted into. Second, if larger polyhedra than the eight

polyhedra given by Conway are needed, no method is given to construct these and because of the lacking symmetry it is difficult to determine them. We would like to remedy these drawbacks by changing the notation slightly.

We will give a basic polyhedron, which we will call the *universal polyhedron*, that can be scaled up or down very easily as needed. It is much easier to insert a knot into the universal polyhedron because of its internal symmetry and also because we only need elementary tangles and not rational tangles as we did before. We are able to show an upper bound on the size of the universal polyhedron for a specific crossing number so that we do not use an unnecessarily large version of the polyhedron to denote a knot.

After establishing these basic properties, we will find that the structural symmetry and scaling properties of the universal polyhedron allow us to give a general algorithm for generating a closed braid (or closed plait) representative of the knot. Finally, we are able to show that this braid generation algorithm is, in some sense, better than existing algorithms.

3.1 The Universal Polyhedron

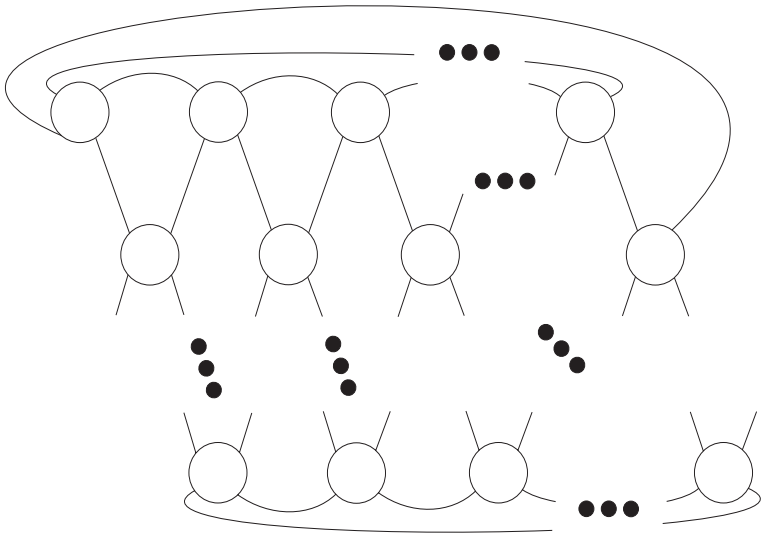

Figure 4: The polyhedron $P(i, j)$

The universal polyhedron, denoted by $P(i, j)$ is shown in figure 4. It is a prototype for a knot projection. The circles will be called *vertices* and the lines connecting them *edges*. The vertices are arranged into i rows of j vertices each. Each vertex can thus be labelled by its row and column index. Each vertex is connected to four of its neighbors. Usually these are the neighbors (1) above, (2) above right, (3) below and (4) below left except on the boundaries of the

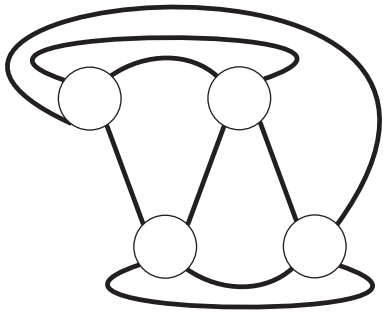

Figure 5: The polyhedron $P(2,2)$

polyhedron. We draw a single arc directly above and below the collection of vertices to connect the extreme vertices of these two rows. All other arcs go on top of this diagram to connect a row to the row below. To be specific, the polyhedron $P(2,2)$ is shown in figure 5.

While $P(i,j)$ denotes the whole polyhedron and specifies the number of rows and columns, p_{kl} specifies a particular vertex in row k and column l. In what follows, we will substitute rational tangles into the vertices to yield a knot projection. Since a rational tangle may be specified by a single extended rational number, p_{kl} takes an extended rational number value. This can be completely specified by giving all p_{kl} a value, which may be arranged into a matrix form,

$$P(i,j) = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1j} \\ p_{21} & p_{22} & \cdots & p_{2j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i1} & p_{i2} & \cdots & p_{ij} \end{pmatrix} \tag{7}$$

For any knot K, if we have determined a matrix $P(i,j)$ that gives rise to a knot projection of a knot ambient isotopic to K, we call this matrix the *knot matrix* of K. Interpreted in the above manner, it is clear that any matrix with extended rational numbers entries gives rise to a knot. Since this is true for rational tangles, it is true for any subset of the rational tangles, in particular the elementary tangles. Thus if all p_{kl} take a value from the set $\mathcal{E} = \{0, -1, +1, \infty\}$, the result is also a knot projection. We now show that every knot has a knot matrix.

3.2 Any Knot can be Denoted by the Universal Polyhedron

Theorem 1 *Every knot has a knot matrix.*

Proof. We observe that if all vertices take a tangle value of ± 1 or 0, then the polyhedron drawn is the canonical closure of the braid represented by the

vertices and the internal edges of the polyhedron. Alexander's theorem states that every knot may be represented by a closed braid [1]. As every closed n-braid of c crossings may be drawn in the polyhedron $P(n, c)$ by the above observation, we conclude that every knot has a knot matrix. □

From the above discussion, it is clear that two distinct knots have distinct knot matrices and two ambient isotopic knots may or may not have the same knot matrices. Having established that we can always denote a knot in this way, we would like to have a bound on the size of the required universal polyhedron to denote a knot of n crossings.

Theorem 2 *Let K be any knot given by a diagram $D(K)$ with n crossings, then we can always find a square elementary tangle valued knot matrix $P(i, i)$ such that $P(i, i) = D(K)$ with $i \leq 2\sqrt{n}$.*

Proof. Consider an oriented knot diagram $D(K)$ with n double points. Label its double points and edges with the positive integers in the order of its orientation starting at an arbitrary point. We associate to each knot diagram $D(K)$ a graph $G(D(K))$ in the following way. Each crossing point becomes a vertex in the graph and is labelled by the sign of the crossing (a right-handed crossing obtains a label 1 and a left-handed crossing a label -1). We draw a directed edge (i, j) between vertices i and j if and only if there exists a directed arc in $D(K)$ from double point i to double point j. This edge is labelled by the number of the arc in $D(K)$. This gives an edge and vertex labelled 4-regular planar graph $G(D(K))$ with n vertices and $2n$ edges. We note that we may easily reconstruct a knot diagram isomorphic to $D(K)$ from *any* planar drawing of this graph.

Consider the list L of all 4-regular planar graphs on n vertices. If we generate all possible labellings and orientations of each of these graphs, we will obtain every graph arising from a knot diagram of n double points and many others not arising from any knot diagram. All of these graphs have the same vertex set and an edge set of $2n$ edges. The adjacency matrix of any one of these graphs will contain exactly four entries of value 1 in each column and row. All other entries will be zero and the matrix will be symmetric. The list L may be enumerated by generating all possible such matrices. Note that any two matrices which differ by a permutation of the rows and the columns can be obtained from each other by a renumbering of the vertices and thus correspond to an ambient isotopic knot diagram. Such duplicates are to be removed.

We now consider the edges arising from inserting double points into a polyhedron $P(i, i)$. Note that we are focusing on a square polyhedron of side-length i. Clearly all possible ways to fill $P(i, i)$ will generate a 4-regular planar graph. If we add over and under-crossing information and an orientation, we obtain a graph of a knot diagram. Consider the vertex (p, q) in $P(i, i)$. If we change it from a double point to a 0 tangle or to an ∞ tangle, we change the edges in the

following way.

$$
\begin{aligned}
E_0 \to E - &\left\{ \begin{array}{c} \{\{(p,q),(p-1,q)\},\{(p,q),(p+1,q)\}, \\ \{(p,q),(p-1,q+1)\},\{(p,q),(p+1,q-1)\}\} \end{array} \right\} \\
&+ \left\{ \begin{array}{c} \{(p-1,q),(p-1,q+1)\}, \\ \{(p+1,q-1),(p+1,q)\} \end{array} \right\}
\end{aligned} \tag{8}
$$

$$
\begin{aligned}
E_\infty \to E - &\left\{ \begin{array}{c} \{\{(p,q),(p-1,q)\},\{(p,q),(p+1,q)\}, \\ \{(p,q),(p-1,q+1)\},\{(p,q),(p+1,q-1)\}\} \end{array} \right\} \\
&+ \left\{ \begin{array}{c} \{(p-1,q),(p+1,q-1)\}, \\ \{(p-1,q+1),(p+1,q)\} \end{array} \right\}
\end{aligned} \tag{9}
$$

These formulae hold except at the boundaries where we must take the numbers mod i and in addition take care of the bottom and top line. This operation thus decreases the number of crossings by one and the number of edges by two. These replacement formulae also make apparent that the 0 tangle represents a horizontal connector and the ∞ tangle a vertical connector between crossings.

We see that if we start from a suitably large polyhedron we can get to any given graph $G(D(K))$ by such replacements taking care that we get the edges in the given edge set. We note that since edges are deleted and added in pairs by this process, we need one row and one column in between every neighboring pair of crossings. By making such replacements and by permuting the crossings among each other, we can clearly form any adjacency matrix in the above list. The polyhedron matrix looks like the configuration given in equation 10,

$$
\begin{pmatrix}
\times & \square & \times & \square & \cdots \\
\square & \square & \square & \square & \cdots \\
\times & \square & \times & \square & \cdots \\
\vdots & \vdots & \vdots & \ddots & \vdots
\end{pmatrix} \tag{10}
$$

where an empty box is to be understood to be filled by a 0 or ∞ tangle and a cross is a ± 1 tangle. This configuration with $2l$ rows and $2l$ columns has l^2 crossings. Thus we conclude that the polyhedron $P(2l, 2l)$ can contain any knot diagram of l^2 crossings up to Reidemeister moves. This proves the theorem. □

Note that theorem 2 was proven for elementary tangles substituted into the vertices of the universal polyhedron. We find in practise that by using non-elementary rational tangles the size of the polyhedron can be reduced significantly. Furthermore, we note that in practise it is rare for a polyhedron of the maximal size proven to be necessary.

3.3 Finding the Knot Matrix

We wish to find the knot matrix for a given knot projection. Since this is an algorithmic question, we must ask in what fashion the knot is already given. By

Alexander's Theorem, every knot can be represented by a closed braid. If we have the knot given by its closed braid representative, it is trivial to put it into the universal polyhedron as the polyhedron is nothing more than braid closure if the tangles take on the values ± 1 and 0. If the knot is given in Conway's notation, translation formulae are given in the appendix of this paper. Both of these methods of finding knot matrices can be completely automated.

Generally, however, a knot is given by one of its projections onto the plane. By the above observations, we could simply execute Alexander's [1] or Vogel's [16] algorithm to find a closed braid representative of the knot projection and place this in the universal polyhedron. This unnecessarily increases the crossing number however and makes the use of a large polyhedron necessary.

We note that by theorem 2, we could enumerate all possible polyhedra satisfying the bound and test via graph isomorphism whether the generated knot matrix gives the knot at hand. This algorithm is simple and will terminate in a very small knot matrix. It is however not practical as the list includes an exponentially growing number of members.

In addition to these methods of producing a knot matrix, we add another below which produces a knot matrix without increasing the crossing number and using a reasonably sized polyhedron. This method requires the production of the matrix associated to the knot diagram. After this has been done by a human, the rest of the algorithm may be done by a computer.

Algorithm 1 *Input: A knot diagram $D(K)$ with n crossings. Output: A knot matrix for the given projection.*

1. Obtain $G = G(D(K))$ and construct an i by i knot matrix M where $i = \lfloor 2\sqrt{n} \rfloor$. Fill all the entries with both an odd row and odd column index with a 1 tangle and leave all the others empty. Number the vertices across the rows and down the columns in order.

2. Fill the empty entries in M with 0 and with ∞ tangles in such a way that the corresponding edges to be added to the graph G' associated to M are a subset of the edge-set of G. The fact that this is possible is stated by theorem 2.

3. The previous step has created a knot matrix whose associated graph is isomorphic to the unlabelled and undirected version of G. By the constructed isomorphism, we now add the orientation and labelling to G'. The end result will be a knot matrix for the knot K.

3.4 Getting Information about the Regions of the Plane

We will now assume that the knot of interest is given in our matrix notation with $p_{kl} \in \mathcal{E}$. We will now give an algorithm that will give us information about the regions into which the knot diagram divides the plane. The number of regions into which the knot projection partitions the plane is important in a

few applications such as the braiding algorithm that is described later in this paper.

We distinguish between *regions* of the plane and *sections* of the polyhedron. By regions of the plane, we mean those regions into which the knot projection divides the plane. By sections, we mean those regions into which the universal polyhedron *would* divide the plane if all vertices were filled with a 1 tangle. Each vertex has exactly one section lying to its right in the polyhedron and thus we may label all these sections by the row and column indices of the associated vertex. Only two sections are not indexed by this method, these are the two sections with j vertices directly on the top and bottom of the matrix construction. These will be labelled by the pairs $(0, 1)$ and $(j + 1, 1)$. Thus the sections may be represented by a matrix. If the entries of this matrix are made to take integer values we may count the number of regions by the following algorithm.

Algorithm 2 *Input: A knot matrix. Output: A matrix describing the regions.*

1. We need a counter, k to count the regions. Initialize k to 1. We have a matrix S_{pq} which will store the number of the region that the section (p, q) belongs to. A 0 or an ∞ tangle connects two sections and thus they belong to the same region. Each position in the matrix S belongs to one section so that in the end the matrix S gives complete information which region each section belongs to and in particular how many regions there are. Initialize all elements of S to "unmarked" and $p = 0$ and $q = 1$.

2. Begin on the boundary of section (p, q) at the far left. For counting regions, the orientation of the knot does not matter, therefore follow the boundary to the right. Put $S_{pq} = k$.

3. We follow the boundary of the sections marking each new section with the integer k.

4. We continue to follow the boundary until we reach the point of origin.

5. We search the matrix for an unmarked section. If there exist unmarked sections, put $k \leftarrow k + 1$ and choose one of the sections as our new starting section and choose a point upon its boundary as our new starting point. Then, we repeat the algorithm from step 2, marking the section with k.

6. If all sections are marked, the algorithm is finished. k is clearly the number of regions. Note that the infinite section is included in this labelling as section (i, j). Furthermore, since all connected sections are labelled with the same integer, we have a complete knowledge of where the regions lie.

The algorithm considers each vertex exactly twice and therefore the complexity is $O(n)$.

3.5 Getting Information about the Components of the Knot

This new notation can be readily used in calculating some invariants of the knot. For example, to calculate a polynomial invariant for which we have a state model (the Jones polynomial for example [9]), we simply replace each ± 1 tangle by a 0 or ∞ tangle in all possible ways to yield all possible states of the knot. If a knot has n double points, this means 2^n states. We associate an algebraic factor with the way in which this replacement is made and then multiply it by an algebraic factor depending on the number of unknots left. All these contributions are added and yield a polynomial invariant of the knot. The key is to be able to calculate the number of remaining unknots. We now give an algorithm to calculate the number of components in a knot given in our notation. This would provide the number of remaining unknots. By combining the bounds on the size of the polyhedron and this calculation, it is clear that our notation is suitable for the enumeration of knots and detection of duplicates in the enumeration. It could thus be used to find a table of all distinct prime knots up to a certain number of crossings.

Algorithm 3 *Input: A matrix describing a knot in our notation. Output: A matrix giving information about the location and number of the components of the knot.*

1. Each vertex has four points in which the two polygonal curves intersect the 3-ball that denotes the volume taken by the tangle. These are shown in figure 1 and named NW, SW, NE and SE. We initialize a matrix C_{ijk} where i and j give the vertex and k counts over the four points NW, SW, NE and SE at that vertex. We put $C_{ijk} = 0$ for all i, j and k. We need a counter m that will count the components. We initialize $m = 1$. We initialize the current point to be $i = j = 1$ and $k = NW$, i.e. we start at the extreme top left of the universal polyhedron.

2. Start at point k of the vertex (i, j) by putting $C_{ijk} = m$.

3. We follow the orientation and not the boundary, as in algorithm 2, marking each point on each vertex as we pass it in the matrix C with the counter m.

4. When we reach the point of origin, we put $m \leftarrow m + 1$ and look for an unmarked point (some $C_{ijk} = 0$).

5. If there is an unmarked point, we take it as the current point and proceed. We loop from step 2 to 5 until all points are marked.

This method calculates the number of components considering each point on each vertex once, therefore the complexity is also $O(n)$. Note that a matrix of only 0 tangles contains $i + 1$ unknots and a matrix composed of only ∞ tangles contains j unknots.

Smaller polyhedra $P(i,j)$ may be embedded in larger ones by filling in the rest with 0 and ∞ tangles. Conversely, if the configuration of the tangles is right, we may delete rows and columns accordingly. For example, we may create an extra row at the bottom or top of the matrix containing

$$(0 \quad 0 \quad \cdots \quad 0 \quad \infty)$$

and we may add an extra column at the left or right of the matrix containing only 0 tangles. Likewise, such columns may be removed without changing the knot type. Thus if a given knot can be expressed in the polyhedron $P(i,j)$ it can also be expressed in any polyhedron $P(i',j')$ for which $i' \geq i$ and $j' \geq j$. An internal row of 0 tangles splits the polyhedron into two parts each described by the matrix above and below the row of zeros. Thus if two knots should be described in a single diagram without touching, this is a way in which this may be done.

4 Braiding a Knot

The motivating problem to start this research was to develop an implementable algorithm to produce a closed braid ambient isotopic to any given knot. Some algorithms exist but they all depend on topological deformations of some kind and so are not immediately implementable [10] [3]. The best known algorithms have been implemented [16] [17] but have complexity $O(n^2)$ for an n crossing knot projection. They also give rise to braids with many more crossings than necessary.

We shall present an algorithm that uses less crossings than any known algorithm, will achieve its task by modifying the knot matrix and is immediately implementable. It will also achieve the conversion with complexity $O(n)$, increase the number of crossings from n only sometimes (and then by only a few crossings) and use a linearly bounded number of strings. There exists no algorithm to calculate the braid index of the knot, i.e. the number of strings which are at least necessary to describe a specific knot. Because of this, it is not possible to say how close to the minimum the number of strings used is.

We note that there exist knots for which *any* closed braid representative has more crossings than the minimal knot diagram; the knot 5.1 is the simplest example of this [13]. If we are given the minimal crossing knot diagram of such a knot, we must therefore increase the number of crossings in producing a closed braid representative. A bound on this necessary increase is not known and so we can not say how close to the minimum we are.

4.1 Closures of Braids

We have the notational armory now to discuss an algorithm to transform a knot into a closed braid form. That this is in principle possible is assured by Alexander's Theorem [1] which states that any knot may be expressed as a closed braid. There are two popular ways to close a braid: the canonical closure

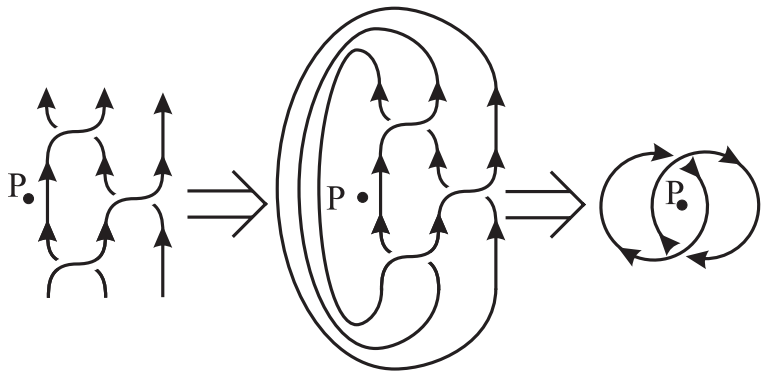

Figure 6: The (canonical) closure of a braid. In the braid group language, the braid is $\Delta_3 = \sigma_1\sigma_2\sigma_1$ and the knot is the Hopf link.

and the plait closure. In the canonical closure, which is used more often, one numbers the strings in the braid on both the top and bottom consecutively from left to right and connects the strings with the same number. In the plait closure, one numbers the strings in the same way but connects each odd numbered string to the string with the next even number. It is intrinsic to the definition of the plait closure that the number of strings must be even. It is because of this severe limitation that the plait closure is used infrequently; the literature reserves plait closure almost exclusively for 4-braids. Both closure methods are illustrated in figures 6 and 7.

Alexander's theorem was proven by showing that every knot can be deformed into a form where the knot loops around an axis (shown as point P in figure 6) 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 the given 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. Before we proceed to illustrate the general algorithm, we give an example in the next section.

4.2 An Example

For the rest of this section, we are going to work through an example of our method. Consider the trefoil knot in figure 8. 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

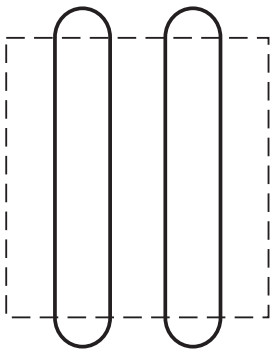

Figure 7: The plait closure of a braid. Note that there is potential conflict between orientations of the braid strings in the plait closure; it becomes impossible to plait a braid in which all strings are oriented in the same way.

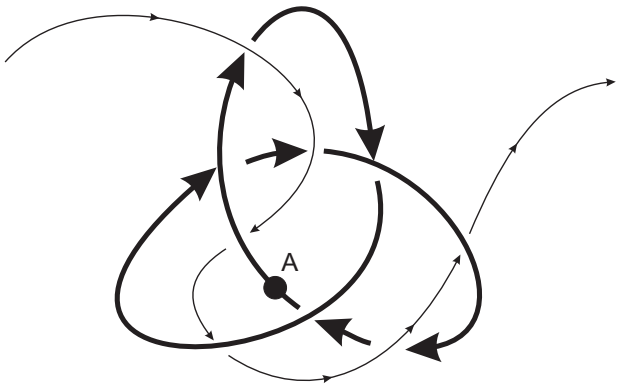

Figure 8: The trefoil knot with an axis for braiding it.

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 8) to the axis reference frame in figure 9 by pulling the axis straight.

We can easily observe from figure 9 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 over-crosses 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 10 (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 10 (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

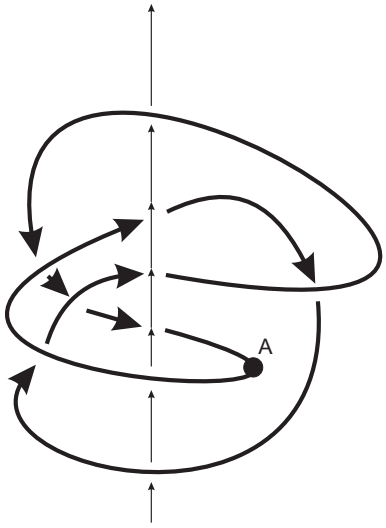

Figure 9: The trefoil knot as it appears after the axis has been straightened from figure 8. For reference the point A has been labelled here again.

which the most efficient braid representation contains more crossings than their most efficient knot projection [13]). We note that we may lift the arc labelled in figure 10 (b) to remove one crossing. This move also removes a string and so we obtain the braid of figure 10 (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 10 (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.

4.3 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 important feature of the braid part, for our purpose, is the requirement that all strings be monotonic increasing in the coordinate along the axis, 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

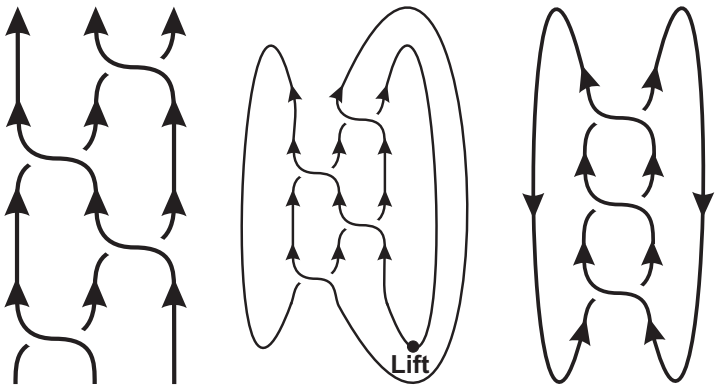

Figure 10: The braid which is extracted from figure 9 by cutting the trefoil knot at its over-crossings over the axis and laying out the ends is displayed in part (a). The closure of this braid is part (b). If we lift the arc labelled in part (b) we obtain the braid in part (c). See discussion in the text.

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 regularly isotopic to the knot we started with. This is hard to visualize and so we give an example of this algorithm.

Figure 11 shows the process of converting the unknot

$$U = \begin{pmatrix} -1 & 1 \\ \infty & -1 \end{pmatrix} \tag{11}$$

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

4.4 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

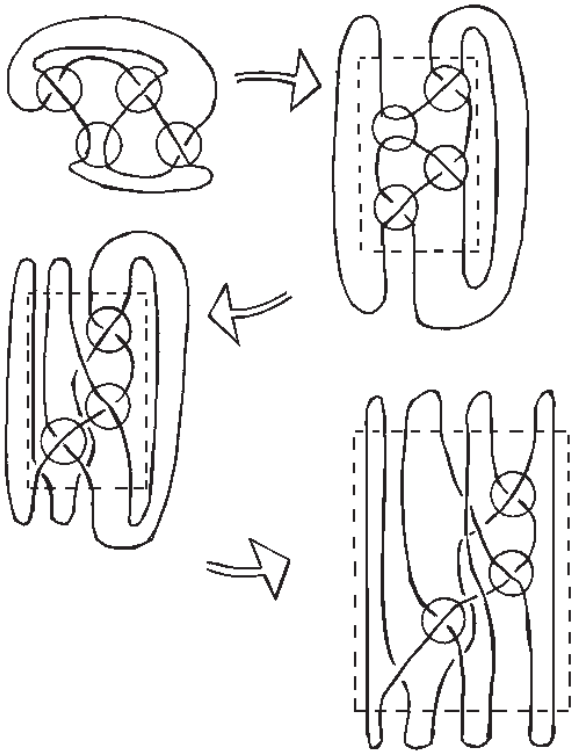

Figure 11: The conversion of a knot into a plait.

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 4 *Input: A knot projection given in our notation. Output: A knot projection with an axis around which the knot winds without local maxima or minima.*

1. Run algorithms 2 and 3 to get the information about the regions and components. Suppose that there are R regions and C components.
2. Choose two arbitrary points in the infinite region and call them A and C.
3. Draw a line L connecting A and C 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 C is then a valid axis.

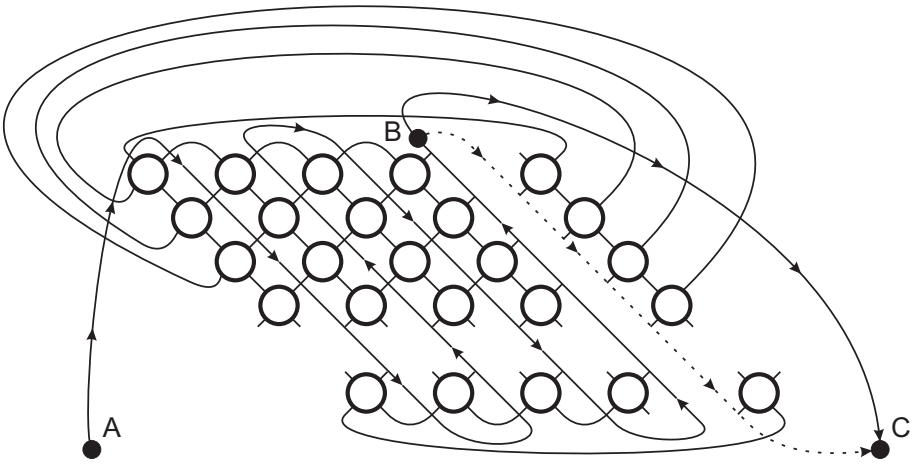

Figure 12: The canonical axis of the braid through the polyhedron $P(i, j)$. The solid line applies when j odd and the dashed line is the short-cut that applies when j is even.

It is clear that the algorithm may be applied to any polyhedron. We will prove shortly that the line found is always a valid axis. Before we prove this,

however, we draw attention to the crucial step of the algorithm, namely step 3. This can done in various ways. One way is to draw a line over the polyhedron such that the line intersects each section of the polyhedron at least once. Such a line is shown in figure 12. This will clearly intersect each region at least once and has the advantage that it is a canonical line that does not need to change from knot to knot. This line is effectively hard-coded into the polyhedron and will be the line we shall accept for practical use.

It is possible to minimize the number of crossings of the line with the knot by solving the Hamiltonian cycle problem for a graph that has a node for each region and an edge between adjacent regions. As this is currently only possible in exponential time, we do not suggest this route. One may use heuristics for this problem to achieve the minimum in most cases. We note at this time that the minimization of intersections between the line and the knot is important because it will turn out that the number of strings in the braid is equal to one-half times the number of these crossings. We will find that with the canonical line, we can calculate this number and it is linear in the number of crossings of the knot. This is clearly an upper bound for the method focusing on Hamiltonian paths. Having stated this caveat, we prove that algorithm 4 always yields a valid axis, this essentially amounts to proving Alexander's theorem.

Theorem 3 *Given any knot matrix, algorithm 4 will find an axis about which the knot is without local maxima or minima.*

Proof. Alexander's theorem [1] states that 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 [12] 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 while 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 under-cross the knot alternately with respect to A, i.e. the axis must alternately over and under-cross the knot as seen by a point traversing the knot. This is precisely the axis found in the algorithm and thus proves the theorem. □

4.5 Getting the Braid

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

Algorithm 5 *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 regularly isotopic to the given knot.*

1. Consider an empty polyhedron $P(i, j)$. Each edge can be thought of emerging from the East side (remember that we labelled the endpoints of the tangles NW, NE, SW and SE) of exactly one vertex. We therefore label each edge by the triple (i, j, k) where i and j are the column and row indices of the vertex and k is either NE or SE. So we represent the edges as another matrix. Choose a point A on the knot.

2. All edges which under-cross the axis L are to be numbered in order starting at point A and proceeding in the order reached if the knot is traversed starting at A. Suppose there are k of these.

3. For each numbered edge, follow each edge around the knot until another edge under-crossing L is reached. 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.

5. Traverse the knot again starting at point A and extracting which labelled string passes over which other labelled string at each vertex containing a ± 1 tangle.

6. The result is a list of crossings between strings "colored" with the colors $1, 2, \cdots, k$. Thus the output of the previous step is the braid in its colored braid group representation. This can be converted into an Artin braid word easily. If a colored generator requires two non-adjacent strings to cross, the top string must first move behind all the intermediate strings before the required crossing can be realized.

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. In order to maintain isotopy, the permutation braid which is appended must be entirely composed of inverse generators.

The number of crossings is increased in some circumstances by a small amount in step 7 of the algorithm. Steps 6 and 7 is not a deficiency of algorithm 5 but a fundamental necessity as mentioned previously. Crossings are

only added when it is necessary for the permutation of the braid. The fundamental braid word $\Delta_n = \sigma_1\sigma_2\cdots\sigma_{n-1}\sigma_1\sigma_2\cdots\sigma_{n-2}\cdots\sigma_1\sigma_2\sigma_1$ of length $n(n-1)/2$ represents the maximal permutation and so the number of crossings is increased by $n(n-1)/2$ at most. It is furthermore possible, by the way the extra crossings are computed, that some crossings may be cancelled over the free group or by more sophisticated reduction methods [2].

It is clear from Alexander's theorem [1] 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

$$N_c = \begin{cases} (i+1)j & j \text{ even} \\ (i+1)j + i - 1 & j \text{ odd} \end{cases} \tag{12}$$

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. Therefore, the number of strings of this braid is $N_c/2$ which scales linearly with n.

Algorithm 5 therefore finds a braid with a number of strings which scales linearly in the size of the knot matrix. The number of strings may be reduced after the braid has been found using Markov's theorem, even though no algorithm for such a reduction exists.

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. The building of the braid from the crossing information takes time proportional to n. Therefore the entire algorithm to proceed from a knot projection to a canonically closed braid has complexity $O(n)$. This algorithm succeeds in being readily implementable and in constructing a braid which is reasonably small. It is more efficient both in computing time and crossing number than the best known algorithm by Vogel [16] which runs in $O(n^2)$. This algorithm increases the crossing number by $n^2 - 3n + 2$ at most but uses a number of strings equal to the number of Seifert circles in the knot projection. Thus while our algorithm is faster and produces a braid with less crossings, it nevertheless produces a braid with more strings than Vogel's algorithm. The translation algorithm has been implemented by the author as part of a larger knot theory program called BraidLink which is available from the author.

A Translating from Conway's notation to the universal polyhedron

If the knot is given in Conway's notation [5], we may make the translation into our notation by fitting the appropriate basic polyhedron into $P(i,j)$ with a specific choice for i and j. Since Conway uses integral tangles for his notation,

this method will yield a matrix with integer number entries. In the equations below we write Conway's notation and ours in correspondence, the letters imply integral tangles and the operator M is to be understood as the mirror reflection operator. The equations may be verified readily by drawing the diagrams, they have been omitted here for reasons of space.

$$1^*a = (a) \tag{13}$$

$$6^*a.b.c.d.e.f = \begin{pmatrix} a & c & e \\ M(b) & M(d) & M(f) \end{pmatrix} \tag{14}$$

$$6^{**}x.a.b.c.d.y = \begin{pmatrix} \infty & a & x \\ \infty & M(b) & \infty \\ \infty & c & \infty \\ M(y) & M(d) & \infty \end{pmatrix} \tag{15}$$

$$8^*a.b.c.d.e.f.g.h = \begin{pmatrix} a & c & e & g \\ M(b) & M(d) & M(f) & M(h) \end{pmatrix} \tag{16}$$

$$9^*a.b.c.d.e.f.g.h.i = \begin{pmatrix} a & d & g \\ M(b) & M(e) & M(h) \\ c & f & i \end{pmatrix} \tag{17}$$

$$10^*a.b.c.d.e.f.g.h.i.j = \begin{pmatrix} a & c & e & g & i \\ M(b) & M(d) & M(f) & M(h) & M(j) \end{pmatrix} \tag{18}$$

$$10^{**}a.b.c.d.e.f.g.h.i.j = \begin{pmatrix} a & c & e & 0 & 0 & \infty \\ M(b) & M(d) & M(f) & M(h) & M(j) & \infty \\ 0 & 0 & g & i & 0 & \infty \end{pmatrix} \tag{19}$$

$$10^{***}x.a.b.c.d.e.f.g.h.y = \begin{pmatrix} \infty & 0 & a & x \\ \infty & M(e) & M(b) & \infty \\ \infty & f & c & \infty \\ \infty & M(g) & M(d) & \infty \\ y & h & 0 & \infty \end{pmatrix} \tag{20}$$

$$11^*a.b.c.d.e.f.g.h.i.j.k = \begin{pmatrix} a & c & e & g & i \\ M(b) & M(d) & M(f) & M(h) & M(j) \\ 0 & 0 & \infty & 0 & \infty \\ 0 & \infty & M(k) & 0 & \infty \end{pmatrix} \tag{21}$$

Using integral tangles in our matrix notation makes the notation more compact in that fewer rows and columns are needed to denote a knot but it also makes it more complex since each matrix element may take many values. We have described algorithms which assume that the matrix only takes on elementary tangle values, therefore we must have a method for separating out the integral tangles introduced into the notation via equations 13 - 21.

Algorithm 6 *Input: A matrix describing a knot in our notation in which one or more elements are integral tangles. Output: A matrix describing the same knot in which all elements are elementary tangles.*

1. Focus attention on the first integral tangle which is not elementary, suppose this has value sk where $s = \pm 1$ is the sign and k is a positive integer greater than one.

2. Create $k-1$ columns immediately after the column containing the current integral tangle and fill each new vertex with a 0 tangle.

3. Suppose the current integral tangle is $p_{mn} = sk$. Then set $p_{mq} = s$ for $n \leq q \leq n + k - 1$.

4. Finally exchange the values of elements $p_{m+1\ n}$ and $p_{m+1\ n+k-1}$.

It is easy to convince oneself, by drawing a few diagrams, that algorithm 6 will accomplish the decomposition. While the number of rows stays constant, the number of columns may explode if there are numerous integral tangles of high values in the matrix. However this algorithm will give us a ready means, together with equations 13 - 21, to convert any knot given in Conway's notation into our notation.

References

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

[2] P. D. Bangert and M. A. Berger, *In Search of Minimal Random Braid Configurations*, J. Phys. A **35** (2002) 43 - 59.

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

[4] G. Burde and H. Zieschang, *Knots*, Walter de Gruyter, Berlin (1985).

[5] J. H. Conway, *An Enumeration of Knot and Links, and Some of their Algebraic Properties*, ed. J. Leech, Pergamon, Oxford (1970) 329 - 364.

[6] C. H. Dowker and M. B. Thistlethwaite, *Classification of Knot Projections*, Topology Appl. **16** (1983) 19 - 31.

[7] I. Fary, *On Straight Line Representations of Planar Graphs*, Acta. Sci. Math. Szeged **11** (1979) 837 - 863.

[8] J. R. Goldman and L. H. Kauffman, *Rational Tangles*, Adv. Appl. Math. **18** (1997) 300 - 332.

[9] L. H. Kauffman, *Knots and Physics*, World Scientific, Singapore (2001).

[10] S. S. F. Lambropoulou, *A Study of Braids in 3-manifolds*, Ph. D. Thesis, University of Warwick (1993).

[11] K. Murasugi, *Knot Theory and Its Applications*, Birkhäuser, Boston (1996).

[12] H. R. Morton, *Threading Knot Diagrams*, Math. Proc. Camb. Phil. Soc. **99** (1986) 247 - 260.

[13] R. L. Ricca, *Applications of Knot Theory in Fluid Mechanics*, ed. V. F. R. Jones, Banach Centre Publ., **42**, Polish Acad. Sci., Warszawa (1998) 321 - 346.

[14] D. W. Sumners, *Untangling DNA*, Math. Intelligencer **12** (1990) 71 - 80.

[15] M. B. Thistlethwaite, *Knot Tabulations and Related Topics*, LMS Lecture Notes No. 93, Cambridge University Press, Cambridge (1985) 1 - 76.

[16] P. Vogel, *Representations of links by braids: A new algorithm*, Comment. Math. Helvetici **65** (1990) 104 - 113.

[17] S. Yamada, *The minimal number of Seifert circles equals the braid index of a link*, Invent. Math. **89** (1987) 347 - 356.