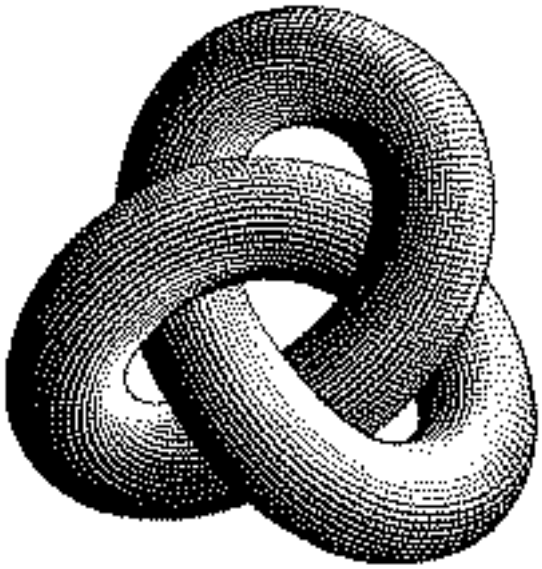


Knot classification and enumeration

Nicholas James Jackson
Supervised by Dr Brian Sanderson

30th September 1997

Joseph Leslie Slack (1908-1996)

Wilfred Leslie Jackson (1932-1997)

Contents

List of Figures

Chapter 1

Introduction

This dissertation is based mostly on John Conway's 1970 paper *An enumeration of knots and links and some of their algebraic properties* [5] and follows, broadly, the structure of that paper. I have attempted to expand and expound upon the material therein, referring in many cases to other papers or books which are listed in the bibliography.
It should be stated that none of the material in this dissertation, with the possible exception of the C++ sourcecode in the appendix, is original work. My aim is to provide an expository work examining a number of topics in classical knot theory.

1.1 Knots, links and diagrams

This section is intended to serve as a very brief introduction to the mathematical theory of knots, defining some basic concepts and terminology. More thorough introductions may be found in [4], [17] and [9] amongst others. Some more traditional (non-mathematical) texts on knots include [2], [8] and [10].
A **knot** is an embedding $K : S^1 \to S^3$ and a **link** is an embedding $L : \coprod_{i=1}^{n} S^1 \to S^3$, that is, a finite number of (possibly linked) knots. In practice, the distinction between the embeddings and their images is often not made, and the terms 'knot' and 'link' will be used to refer to either unless stated otherwise.
A knot K is usually depicted by means of a projection $p : K \to \mathbb{R}^2$. A point $P \in p(K)$ is said to be a **multiple point** if the preimage $p^{-1}(P)$ is not a single point. A projection is **regular** if it has only a finite number of multiple points, all of which are double points, and no vertex of K is mapped onto a double point. The minimal number of double points in a regular projection of a knot K is called the **order** or **crossing number** of K. The image $p(K)$ is called a **diagram** for K.
Two embeddings $f_0 : X \to Y$ and $f_1 : X \to Y$ are **isotopic** if there is an embedding $F : X \times I \to Y \times I$ such that $F(x,t) = (f(x,t),t)$ for $x \in X$ and $t \in I = [0,1]$ with $f(x,0) = f_0(x)$ and $f(x,1) = f_1(x)$. F is called a **level-preserving isotopy** between f_0 and f_1.
Two embeddings are said to be **ambient isotopic** if there is a level-preserving isotopy $H : Y \times I \to Y \times I$ with $H(y,t) = (h_t(y),t)$ such that $f_1 = h_1 f_0$ and $h_0 = \mathrm{id}_Y$. H is called an **ambient isotopy**.
Two knots, K_1, K_2 are then said to be **equivalent**, or of the same **knot type**, denoted $K_1 \approx K_2$ if they are ambient isotopic.
Two knot diagrams D_1, D_2 are said to be **equivalent**, written $D_1 \sim D_2$, if one can be transformed into the other by an ambient isotopy of the plane, and a finite sequence of **Reidemeis-**

ter moves, as depicted in figure 1.1.

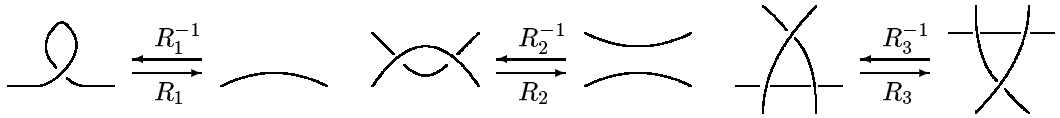


Figure 1.1: Reidemeister moves

Theorem 1.1 (Reidemeister 1932)
Given two knots (or links) K_1 and K_2 with diagrams D_1 and D_2 respectively, K_1 is ambient isotopic to K_2 iff $D_0 \sim D_2$.

1.2 Acknowledgements

I would like to thank my supervisor, Dr Brian Sanderson, for his help and encouragement, and for introducing me to the mathematical theory of knots. I would also like to thank my fellow graduate students for interesting discussions and helpful advice, in particular Michael Greene for his help with proofreading, and Neil Gamble and family for getting me interested in knots in the first place, several years ago.

Chapter 2

Notations

Perhaps the most fundamental question in knot theory, at least historically, is that of classification and enumeration — that is, to devise a method of cataloguing all essentially different knots of a given degree of complexity. This is usually approached by classifying knot diagrams, with a particular number of crossings, up to invariance under the Reidemeister moves and ambient isotopy of the plane. What is required, then, is some concise and consistent notation for describing a knot diagram. In this chapter we describe and examine three different systems of notation.

2.1 Tait's notation

The first serious attempt at an exhaustive enumeration of knots was undertaken by the Victorian physicist Peter Guthrie Tait in the late nineteenth century [20].
Tait's motivation and inspiration for this study came from Kelvin's theory of vortex atoms, which regarded atoms as, in some sense, knots in the vortex lines of the æther. The lack of any detailed correlation between the knot tables compiled by Tait (and his collaborators Kirkman and Little) and Mendeleev's periodic table of the elements was a serious problem for this theory; the coup de grâce was dealt by Michelson and Morley's 1881 experiment disproving the existence of the æther itself.
The fundamental idea behind Tait's system of notation for alternating knots is to label each crossing point in an alternating diagram according to some prearranged algorithm, and then to read off the sequence of labels by following the string round the knot. Each label is then given a subscript $+$ or $-$ to denote whether the crossing is an overcrossing or an undercrossing. This sequence is called a **scheme**.
Tait's method requires one to start at an arbitrary crossing point and follow the string of the knot, labelling every alternate (odd-numbered) crossing sequentially. Since the odd-numbered crossings will always be labelled A, B, ..., the knot in question is then completely determined by the letters in the *even* places in the scheme. For example, the knot in figure 2.1 may be described by the sequence $\mathrm{A}_+\mathrm{D}_-\mathrm{B}_+\mathrm{A}_-\mathrm{C}_+\mathrm{F}_-\mathrm{D}_+\mathrm{B}_-\mathrm{E}_+\mathrm{C}_-\mathrm{F}_+\mathrm{E}_-$, which abbreviates to DAFBCE.
The problem is now that of tabulation of appropriate sequences of letters, subject to certain restrictions. We wish to discount sequences such as EABCD which don't represent a valid knot, by requiring that each label in the scheme occurs exactly twice, as an undercrossing and as an overcrossing. Additionally, we require that if we delete both occurrences of a label L together with both occurrences of any labels between the Ls, the resulting sequence must satisfy the given requirement.

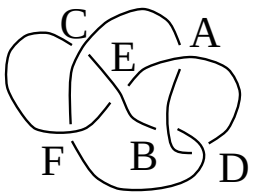


Figure 2.1: Tait's notation for the knot 6_3

As an example of this condition, consider the sequence EABCD, which corresponds to a scheme $A_+E_-B_+A_-C_+B_-D_+C_-E_+D_-$. We may delete the crossing B, and both occurrences of the labels (in this case A and C) which occur between B_+ and B_-, as follows:

A_+	E_-	B_+	A_-	C_+	B_-	D_+	C_-	E_+	D_-
~~A_+~~	E_-	~~B_+~~	~~A_-~~	~~C_+~~	~~B_-~~	D_+	~~C_-~~	E_+	D_-
	E					D		E	D

The sequence EDED, though, does not correspond to a valid knot, however, since attaching $+$ and $-$ subscripts to the labels alternately gives $E_+D_-E_+D_-$, which would require E and D to consist of two undercrossings and two overcrossings, respectively. Hence EABCD cannot correspond to a valid knot.

The reason for this condition is clear: If we eliminate an arbitrary crossing and then discard one of the components of the resulting link, the remaining component must still be a (possibly trivial) knot. If, then, we perform the analogous operation on a sequence which supposedly represents a valid knot, we expect that the resultant sequence still satisfies the fundamental condition — that each label occurs twice, once as an undercrossing and once as an overcrossing. Note that this condition is necessary, but not sufficient to determine whether a given sequence corresponds to a valid knot — for example the sequence EDIHGJBACF satisfies the required condition for every label, but does not correspond to a valid knot diagram.

We may eliminate nugatory crossings as follows: A crossing C is nugatory if, between both occurrences of C in the scheme, there are an even (possibly zero) number of labels, each of which occurs exactly twice.

In enumerating knots of a particular crossing number, then, we may disallow any sequence with A in the second or last place, B in the second or fourth place, C in the fourth or sixth place, and so on, since these give rise to nugatory crossings.

There are essentially four different ways of relabelling a knot in order to derive an equivalent, and possibly different, sequence:

i. We may relabel the crossing B as A, and proceed as usual. This has the effect of replacing each letter by the one preceding it in the alphabet and then moving the first letter to the end. A sequence is said to be **unique** if, as DEABC, it is unchanged under this operation — that is, if the letters are in cyclic order. For a sequence of n letters, the order of the sequence under this operation will be a factor of n. In particular, if n is prime then any such sequence will either be unique or will be reproduced after n such operations.

   For example, considering the knot DAFBCE as depicted in figure 2.1:

A	D	B	A	C	F	D	B	E	C	F	E
F		A		B		C		D		E	

   We now interpolate the appropriate letters (for example, B has been relabelled as A, C as B, and so on) in the odd places to get FCAFBECADBED, or FEABDC.

ii. We may begin from crossing A in the even places and label as before, but by the other

branch of the string. For example:

A	D	B	A	C	F	D	B	E	C	F	E
	F		A		B		C		D		E
A	F	C	A	D	B	F	C	E	D	B	E

Hence, DAFBCE becomes DFEBAC under this operation, which has five other equivalents by the first operation.

iii. We may also reverse the direction of labelling in the first case, so (for example) DAFBCE becomes CEFBAD, which is equivalent to DFEBAC under repeated application of the first operation.

iv. Finally, we may reverse the direction of labelling in the second case, DAFBCE in this case becoming EAFCBD.

2.2 Dowker and Thistlethwaite's notation

Dowker and Thistlethwaite devised a modified form of Tait's notation which has proved particularly suitable for use in computer programs — see for example [1].

For an n-crossing knot, we follow the string round numbering each crossing sequentially. This process constructs a pairing of odd numbers with even numbers, the symbol for the knot in question being the sequence of even numbers $2, \ldots, 2n$ matched with $1, \ldots, 2n-1$. The choice of initial crossing point may result in different symbols for the same knot, so we choose the sequence least in numerical order as the standard symbol for the knot.

We wish only to consider prime knots and hence disallow any sequence such that, for a subinterval of the numbers $1, \ldots, 2n$, the odd numbers in that subinterval are paired with the even numbers in the same subinterval.

In order to reject nugatory crossings, we eliminate any sequence where an odd number is paired with an adjacent even number (mod $2n$). Furthermore, a crossing (i, j) is nugatory if every odd number in the interval (i, j) is paired with an even number in the same interval — sequences of this type will have been discarded, however, as representing composite knots.

As with Tait's notation, not every sequence corresponds to a valid knot, so we further require that any two loops intersect in an even number of points. For example, the sequence 4 6 8 10 2 is invalid since it represents a pairing $\{(1,4),(3,6),(5,8),(7,10),(9,2)\}$, corresponding to two loops $1,2,3,4$ and $5,6,7,8$ which intersect at only one point, namely $(3,6)$. Note that, as with Tait's system, this condition is necessary but not sufficient for determining whether a sequence corresponds to a valid knot.

We adopt the convention that 1 represents an undercrossing, so that, if we consider only alternating knots, the even numbers correspond to the overcrossings of the knot. To extend this notation to alternating knots (which begin to appear when one considers knots of eight crossings and greater), we denote an undercrossing by a minus sign prefixed to the even number in question. The knot 8_{19} (figure 2.2), for example, is represented by the sequence 4 8 -12 2 -14 -6 -16 -10.

2.3 Conway's notation

Conway's motivation for introducing his notation for knots and links [5] was to facilitate computer enumeration of knots and to enable machine calculation of invariants, something which had been hampered until then by the lack of a suitable notation. In the event, such computation was rendered unnecessary by considering various symmetries and other relations between

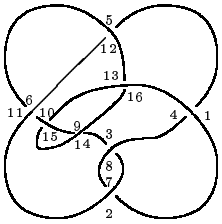
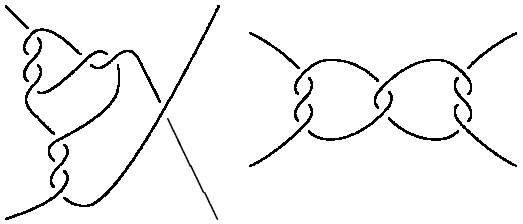


Figure 2.2: Dowker-Thistlethwaite notation for the knot 8_{19}

the knots in his survey, although Trotter [21] used Conway's notation for this purpose with some success, and in chapter 6 we examine its use in a C++ program to calculate invariants for a particular class of knots — the two-bridge knots discussed in chapter 4. Conway's system of notation has the advantage (not entirely shared by either Tait's or Dowker and Thistlethwaite's scheme) that a diagram for the knot or link may be readily obtained from the encoded form.

2.3.1 Tangles

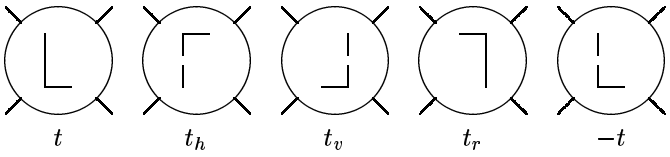
Let a **tangle** be a portion of a knot diagram with four emerging arcs, as in figure 2.3. The northwest arc is called the **leading string** of the tangle, and the northwest-southeast axis is called the **principal diagonal**.



3231 $(3, 2, \bar{3})$

Figure 2.3: Example tangles

An arbitrary tangle may be represented as a circle containing an L-symbol. From this we may obtain fifteen others by various combinations of reflections and rotations, generated by the three operations shown in figure 2.4. Given a tangle t, we denote by t_h and t_v the tangles obtained by rotation about a horizontal (east-west) or vertical (north-south) axis, by $t_r = t_{hv} = t_{vh}$ that obtained by a 180° rotation in the plane and by $-t$ the tangle obtained by reflecting t in the plane of the paper. To visualise this more readily we may regard the tangle symbol as a coin with the L symbol representing the obverse and the broken L representing the reverse.



t t_h t_v t_r $-t$

Figure 2.4: Tangle symmetries

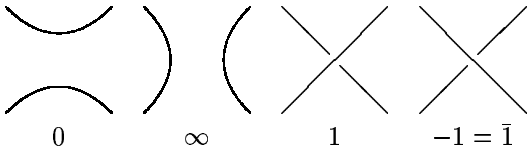


Figure 2.5: Elementary tangles

Figure 2.5 shows the four elementary tangles. Note, however, that Conway's notation follows the opposite of the usual convention by which a *right*-handed crossing is considered 'positive'. Any tangle which may be constructed from the tangles 0 or ∞ by means of the four operations shown in figure 2.6 is said to be **algebraic**.

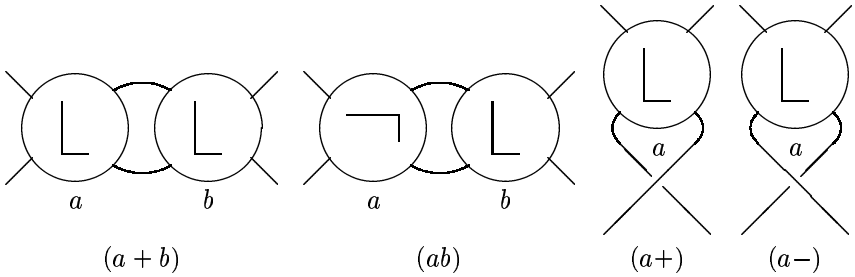


Figure 2.6: Tangle operations

$a + b$ is formed by joining the northeast string of a to the northwest string of b and the southeast string of a to the southwest string of b.

ab is formed by reflecting a in its leading diagonal and joining the northeast string of the resultant tangle to the northwest string of b and the southeast string to b's southwest string.

$a+$ is formed by performing a left-handed twist on the southeast and southwest strings of the tangle a.

$a-$ is formed by performing a right-handed twist on the southeast and southwest strings of the tangle a.

We can define the **integral tangles**, $n = 1 + \ldots + 1$ and $-n = \bar{n} = \bar{1} + \ldots + \bar{1}$, which consist simply of n left- or right-handed twists.
We call a tangle **rational** if it is of the form $((\ldots((ab)c)\ldots)d)$, usually written $abc\ldots d$, the parentheses being associated to the left.
The rational tangles are particularly important — as will be seen later, there is a one-to-one correspondence between the rational tangles and the rational numbers (hence the name). They are also deeply connected with the two-bridge knots and 4-plaits discussed in chapter 4
A rational tangle $a_1 a_2 \ldots a_n$ is essentially the result of starting with the tangle ∞ (if n is even) or 0 (if n is odd) and applying the required number of twists alternately to the southwest and southeast strings, and to the northeast and southeast strings, in such a way that we end up applying a_n twists to the northeast and southeast strings.
Furthermore, the notation $(a, b, \ldots, c)$ is shorthand for $a0 + b0 + \ldots + c0$, but is only used with two or more terms between the parentheses. From consideration of figures 2.5 and 2.6 we see

that $a0$ is equivalent to reflecting the tangle a in its principal diagonal, and that $ab = a0 + b$. Lastly, it should be stated that the notation $a - b$ denotes $a\bar{b}$, rather than $a + \bar{b}$ or $(a-)b$, where $\bar{b}$ denotes the tangle b reflected in the plane of the paper.
Figure 2.7 shows an example of a non-algebraic or **transcendental** tangle, in this case constructed from the polyhedron 8*.

Figure 2.7: A transcendental tangle

2.3.2 Polyhedra

Having constructed a notation for tangles, the next logical step is to extend this to a notation for knots and links. To do this, we introduce the notion of a **polyhedron** — a 4-valent planar graph inscribed on the surface of a sphere. We say a given polyhedron is **basic** if no region has just two vertices adjacent to it. The essential idea is that we replace the vertices of a given polyhedron with algebraic tangles in a clearly-defined, consistent way, in order to construct a knot diagram. The reason for the restriction to basic polyhedra should now become apparent — a region with only two adjacent vertices can be collapsed in a well-defined manner, replacing the tangles at its vertices with a single algebraic tangle which is the sum of the previous two.
To enumerate all knots of eleven crossings or less, and all links of ten or fewer crossings, Conway found that only eight basic polyhedra — depicted in figure 2.8 — were necessary. The six-vertex polyhedron, though, has two forms: 6* and 6**.
The knot formed from the polyhedron P^* by inserting tangles $a, b, \ldots, c$ at the specified vertices is denoted $P^*a.b.\ldots.c$, subject to the following abbreviations:

i. Prefixes 1*, 6* and 6** are omitted, the last of these being denoted by an initial dot.

ii. Tangles of value 1 or -1 are omitted and the separating dots are 'telescoped' — two dots are replaced by a colon and trailing dots are omitted.

iii. The symbol $10-^{***}$ is shorthand for $10^{***}\bar{1}.\bar{1}.\bar{1}.\bar{1}.\bar{1}$.

So, for example, $8^*2 : 3.4 : .5$ is an abbreviation for $8^*2.1.3.4.1.1.5.1$.

2.3.3 Tangle equivalences

Two tangles a and b are said to be **equivalent** (denoted $a \doteq b$) if we can obtain one from the other by performing a finite sequence of Reidemeister moves.
A particularly striking result concerning the rational tangles is that any two tangles $ijk\ldots lm$ and $npq\ldots st$ are equivalent if and only if the corresponding continued fractions $m + \frac{1}{l+\frac{1}{\ddots}}$ and $t + \frac{1}{s+\frac{1}{\ddots}}$ are equal. Justification for this result may be found in chapter 4.
Conway's notation is particularly effective in the sense that it includes Tait's notion of 'flyping' (figure 2.9), in which a tangle $1 + t$ is replaced by $t_h + 1$, or $\bar{1} + t$ by $t_h + \bar{1}$. As will be seen later, for any rational tangle t, we have $t \doteq t_h \doteq t_v \doteq t_r$, so if $a, b, \ldots, c$ are all rational, the position of terms 1 or $\bar{1}$ in a tangle of the form $(a, b, \ldots, c)$ is immaterial and hence the 1 and $\bar{1}$ may be collected together. For example, $(1, 2\ 3, 1, 5) \doteq (2\ 3, 1, 1, 5) \doteq (2\ 3, 5, 1, 1)$. We may further simplify this by using the operations $a+$ and $a-$, since, if $a, b, \ldots, c$ are rational:

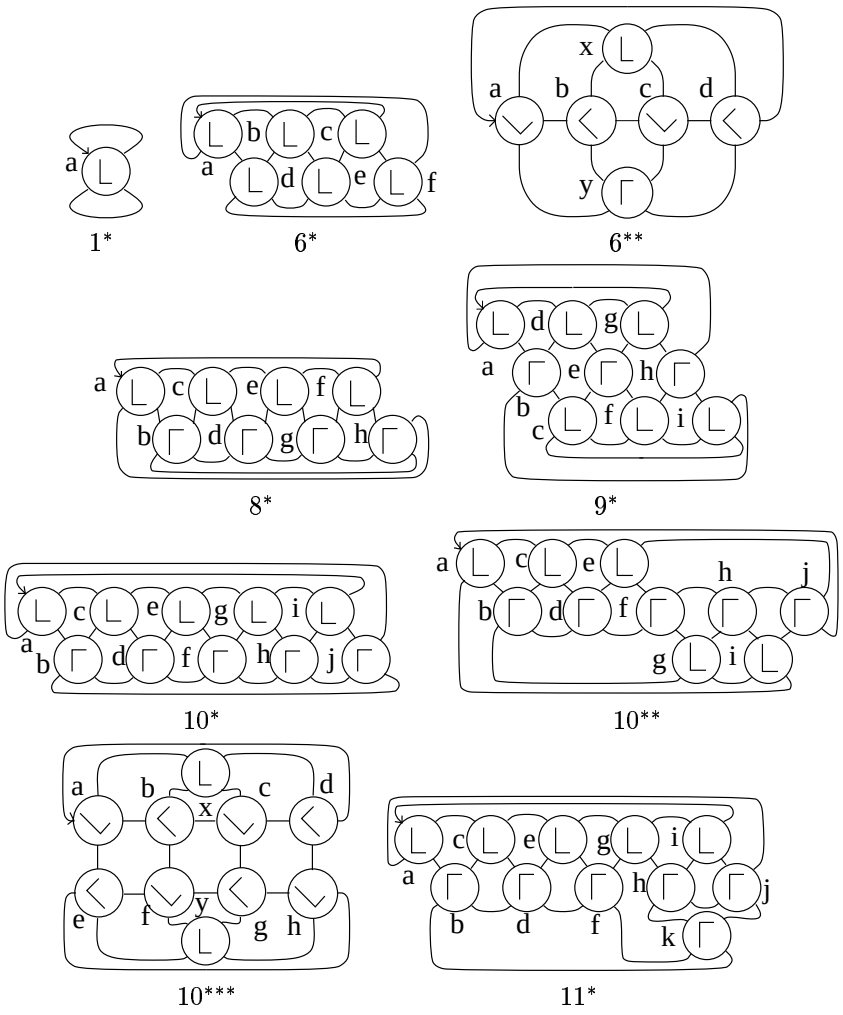


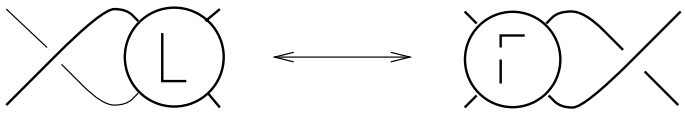
Figure 2.8: Basic polyhedra

$$(a, b, c, 1) \doteq (a, b, c+) \doteq (a, b+, c) \doteq (a+, b, c) \tag{2.1}$$

and

$$(a, b, c, \bar{1}) \doteq (a, b, c-) \doteq (a, b-, c) \doteq (a-, b, c) \tag{2.2}$$

So, we may collect these floating postscripts on the rightmost term, cancelling + and − signs appropriately. If, however, this would leave us with just one tangle t followed by p plus signs and q minus signs then we may further abbreviate the tangle as tn, where $n = p - q$.



$1 + t \longleftrightarrow t_h + 1$

Figure 2.9: Flyping

In the late nineteenth century, Tait suggested the following conjecture which was finally proved by Menasco and Thistlethwaite in 1991 [16]:

Theorem 2.1
Given two reduced, prime alternating diagrams D_1 and D_2 for a knot or link K, it is possible to transform D_1 to D_2 solely by a finite sequence of flypes.

2.3.4 Orientation and string-labelling

We may assign an arbitrary orientation to a given proper knot K by specifying a preferred direction along its string. If this is done, we say the knot is **oriented**. By means of simple geometric operations we may obtain three alternative knots:

i. By reflecting the knot in a mirror we get the **enantiomorph** or **obverse** — Conway's paper denotes this by $\neg K$, other literature tends to denote the reflected knot by K^*.

ii. By reversing the string orientation we obtain another knot — Conway's paper calls this the **reverse** and denotes it by K_r whereas other literature usually calls this the **inverse** and denotes it by K^{-1}.

iii. By reversing the orientation of the reflected knot we obtain a third knot which Conway calls the **inverse**, denoted $\neg K_r$. This is different to what is usually meant by the inverse of a knot, but corresponds to the inverse of K in the cobordism group described in chapter 5.

If $K \approx \neg K$ then K is said to be **amphicheiral** or **achiral**, **chiral** otherwise. If $K \approx K_r$ then K is said to be **reversible** — more usually called **invertible**. If $K \approx \neg K_r$ then K is said to be **involutory**.
The situation is more complex for links of two or more strings, and for this reason Conway introduced an algorithm for labelling and orienting the components of a given knot or link. Orient the leading string of the tangle a in the polyhedron diagrams in figure 2.8 so that the arrow points into the tangle, and label this string r_1. Now follow the string in the direction of its orientation, and label the other strings $r_2, r_3, \ldots$ in order of their first crossing with r_1. Orient each of these strings in such a way that their first crossing with r_1 is a positive crossing in the

sense of figure 2.10. If any strings remain unlabelled and unoriented then repeat this process with string r_2, labelling any unlabelled strings as before, and then with $r_3, \ldots$ if necessary. Note that, as before, this construction is contrary to the usual convention of a right-handed crossing being regarded as 'positive'.

Figure 2.10: Positive and negative crossings

Conway also introduces a notation to describe arbitrary relabelling of strings in a knot or link. Let π be a function (not necessarily a permutation) from the labels $r_1, r_2, \ldots$ to the symbols $r_1, r_1^{-1}, r_2, r_2^{-1}, \ldots$. Then for a given labelled and oriented knot or link K, denote by K_π the knot or link obtained by relabelling all the r_i strings in K as r_j strings in K_π if $\pi(r_i) = r_j$ and as reversely-oriented r_j strings in K_π if $\pi(r_i) = r_j^{-1}$. Define $|\pi|$ to be the number of strings whose orientations are reversed by π and $|K|$ to be the number of components of K. Note that two distinct strings may be given the same label, as is the case with Kauffman's formulation of Conway's potential function.

Chapter 3

Invariants

In this chapter we define and examine a number of knot invariants — that is, a construct (usually numeric or polynomial) derived in a well-defined way from a knot or link, which is unchanged under ambient isotopy. The central requirement, then is that two isotopic links will have equal or equivalent invariants associated with them.

3.1 The Seifert pairing

Given any knot or link K, we may construct a compact orientable surface V whose boundary is K. We may then define a pairing $H_1(V) \times H_1(V) \to \mathbb{Z}$ and derive a number of useful invariants from this.

Proposition 3.1
Any knot K may be considered as the boundary of some compact orientable surface embedded in S^3.

Proof

Consider any diagram D of K. Start at any point of D and move along the knot in the direction of orientation. At each crossing point move onto the other strand and follow that in the positive direction. Eventually, we return to the starting point, having traced out a closed curve (a **Seifert circuit**). Repeat this procedure with all untraversed sections of the knot, constructing a number of disjoint circles. We may consider each of these Seifert circuits to be the boundaries of disjoint discs — if we have a number of concentric circles we may avoid intersections by capping the circuits off with domes. We then join these discs together with twisted bands at the crossing points, as depicted in figure 3.1.
We now have a surface V such that $\partial V = K$. As a consequence of this construction, V is orientable. Each circuit inherits an orientation from the knot, which we may use to attach a normal to each disc using a right-handed screw rule. In passing from one disc to another, the normal is preserved, so there are no closed paths on V which reverse the normal direction. Hence V is orientable. □

This surface V is called a **Seifert surface** for the knot K. We define the **genus** of a surface V to be $g \in \mathbb{Z}$ such that $H_1(V) \cong \oplus_1^{2g}\mathbb{Z}$. We define the genus of the knot K, $\mathbf{g}(K)$, to be the minimal genus over all possible Seifert surfaces for K.

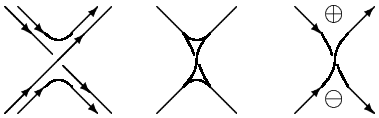


Figure 3.1: The Seifert algorithm

Proposition 3.2
If V is a Seifert surface constructed from some knot diagram, then $g(V) = \frac{1}{2}(c - s + 1)$ where c is the number of crossing points in the diagram and s is the number of Seifert circuits.

Proof

We may construct a handlebody decomposition of V with s 0-handles and c 1-handles, which is equivalent to one with a single 0-handle and $c - (s - 1)$ 1-handles, by cancelling 0-handles. Hence $2g = c - s + 1$. □

Given two oriented, disjoint, possibly linked copies of S^1 embedded in S^3, u, v, we may define the **linking number** of u and v, $\mathbf{lk}(u,v)$ as follows:
Assign an integer ε_i to each crossing point in a diagram of the link such that $\varepsilon_i = +1$ if it is a right-handed crossing, -1 if it is a left-handed crossing, and 0 if the crossing does not involve strings of both u and v. Then define $\mathrm{lk}(u, v) = \frac{1}{2}\sum_i \varepsilon_i$ — half the sum of the εs over all the crossings in the diagram.

Proposition 3.3
$\mathrm{lk}(u, v)$ is independent of the choice of diagram.

Proof

None of the Reidemeister moves change the linking number — R_1 and R_1^{-1} eliminate or create an $\varepsilon = 0$ crossing, R_2 and R_2^{-1} eliminate or create either two 0-crossings, or a positive and a negative crossing which cancel each other out, and R_3 and R_3^{-1} neither create nor eliminate any crossings. Hence the linking number is unchanged by ambient isotopy, and thus independent of the choice of diagram. □

Corollary 3.4
$\mathrm{lk}(u, v) = \mathrm{lk}(v, u)$ for any link components u, v.

Proposition 3.5
$\mathrm{lk}(u, v)$ depends only of the homotopy class of u in $S^3 \setminus v$

Proof

If $u' \simeq u$ in the complement of v, then to pass from a diagram of u, v to a diagram of u', v we have to use the Reidemeister moves and their inverses, together with the crossing change shown in figure 3.1, none of which change $\mathrm{lk}(u, v)$. □

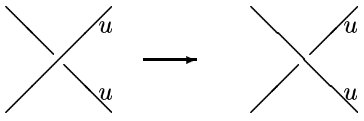


Figure 3.2: Crossing change

Corollary 3.6
$g_v : \pi_1(S^3 \setminus v) \to \mathbb{Z}; u \mapsto \mathrm{lk}(u, v)$ is a well-defined homomorphism.

Let $u, v \in \pi_1(V, *)$ for some Seifert surface V and let i_+u be the result of pushing u a small distance along the positive normal to V. Define $\phi(u, v) = \mathrm{lk}(i_+u, v)$.

Corollary 3.7
If v is fixed then $f_v : \pi_1(V, *) \to \mathbb{Z}; u \mapsto \phi(u, v)$ is a homomorphism.

Since $\mathbb{Z}$ is abelian, $\ker f_v$ contains the commutator subgroup of $\pi_1(V, *)$ and hence f_v induces a homomorphism $H_1(V) \to \mathbb{Z}$. Similarly, we may fix u and vary v to get another homomorphism from $H_1(V)$ to $\mathbb{Z}$, which enables us to define a bilinear pairing $\theta : H_1(V) \times H_1(V) \to \mathbb{Z}$ — the **Seifert pairing**.
We may represent θ by a matrix $A = [a_{ij}]$ where $a_{ij} = \theta(x_i, x_j)$ and $\{x_1, \ldots, x_{2g}\}$ is a basis for $H_1(V) \cong \oplus_1^{2g}\mathbb{Z}$. We say A is a **Seifert matrix**.
Two $n \times n$ integral matrices are said to be **s-equivalent** if one can be transformed into the other by a finite sequence of the following operations and their inverses:

$$\begin{aligned}
\lambda_1 &: A \mapsto TAT^{\mathrm{T}} \\
\lambda_2 &: A \mapsto \left[\begin{array}{c|cc} A & 0 & 0 \\ \hline 0 & 0 & 1 \\ p & 0 & 0 \end{array}\right] \\
\lambda_3 &: A \mapsto \left[\begin{array}{c|cc} A & 0 & q \\ \hline 0 & 0 & 0 \\ 0 & 1 & 0 \end{array}\right]
\end{aligned}$$

where T is a unimodular integer matrix, p is a $1 \times n$ integer matrix, and q is a $n \times 1$ integer matrix.
If two matrices are symmetric then they are said to be **S-equivalent** if one can be transformed into the other by a finite sequence of the following two operations and their inverses:

$$\begin{aligned}
\Lambda_1 &: A \mapsto TAT^{\mathrm{T}} \\
\Lambda_2 &: A \mapsto \left[\begin{array}{c|cc} A & 0 & 0 \\ \hline 0 & 0 & 1 \\ 0 & 1 & 0 \end{array}\right]
\end{aligned}$$

Hence, if M, N are s-equivalent, then $M + M^{\mathrm{T}}$ and $N + N^{\mathrm{T}}$ are S-equivalent.

Proposition 3.8
If M is a Seifert matrix for a link L, then the s-equivalence class of M is an invariant of the link type of L, and hence so is the S-equivalence class of $M + M^{\mathrm{T}}$.

Proof

If two link diagrams represent links of the same type then one may be deformed into the other by a finite sequence of Reidemeister moves, none of which alter the linking number, and hence none of these alters the s-equivalence class of the Seifert matrix M or the S-equivalence class of $M + M^{\mathrm{T}}$. □

3.2 Determinant, signature and nullity

We define the **signature** of a knot or link K, denoted $\sigma(K)$, to be the signature of $M + M^{\mathrm{T}}$, where M is a Seifert matrix for K.

Proposition 3.9
$\sigma(K_1 \# K_2) = \sigma(K_1) + \sigma(K_2)$.

Proof

We can obtain a Seifert surface for $K_1 \# K_2$ by gluing together Seifert surfaces for K_1 and K_2 along a small section of the boundary. If M_1, M_2 are Seifert matrices for K_1, K_2 then $M = \begin{bmatrix} M_1 & 0 \\ 0 & M_2 \end{bmatrix}$ is a Seifert matrix for $K_1 \# K_2$ and hence $\sigma(K_1 \# K_2) = \sigma(M) = \sigma(M_1) + \sigma(M_2) = \sigma(K_1) + \sigma(K_2)$. □

We further define the **nullity**, $\mathrm{n}(K)$, of K to be the nullity of $M + M^{\mathrm{T}}$ and the **determinant**, $\det(K)$ to be $\det(M + M^{\mathrm{T}})$. By proposition 3.8, the determinant, signature and nullity are all invariants of link type.

3.3 The Alexander polynomial

This section is a very brief outline of the Alexander polynomial, an invariant of ambient isotopy for oriented knots. More detailed accounts can be found in [4] and [17].

Let V be a Seifert surface for a knot $K \subset S^3$ and let $N : \overset{\circ}{V} \times (-1,1) \to S^3$ be an open bicollar of the interior $\overset{\circ}{V} = V \setminus K$ of V, so $\overset{\circ}{V} = N(V \times 0)$.
Denote

$$\begin{aligned} N &= N(\overset{\circ}{V} \times (-1,1)) \\ N^+ &= N(\overset{\circ}{V} \times (0,1)) \\ N^- &= N(\overset{\circ}{V} \times (-1,0)) \\ Y &= S^3 \setminus V \\ X &= S^3 \setminus K \end{aligned}$$

Form countably many copies of the triples (N_i, N_i^+, N_i^-) and (Y_i, N_i^+, N_i^-) for $i \in \mathbb{Z}$ and let $\tilde{N} = \coprod_{i=0}^{\infty} N_i$ and $\tilde{Y} = \coprod_{i=0}^{\infty} Y_i$.
We now construct an identification space from these by identifying $N_i^+ \subset Y_i$ with $N_i^+ \subset N_i$, and $N_i \subset Y_i$ with $N_{i+1}^- \subset N_{i+1}$, via the identity homeomorphism. This space, denoted $\tilde{X}$ is the **infinite cyclic cover** of X and is a path-connected open 3-manifold which depends only on the knot type of K and is independent of the choice of Seifert surface V used in the construction.
The homology, $H_*(\tilde{X})$, of $\tilde{X}$ is hence an invariant (the **Alexander invariant**) of the knot type of K and may be regarded as a module over the ring $\Lambda = \mathbb{Z}[t, t^{-1}]$ of Laurent polynomials. In the case of classical tame knots, we only consider the first homology, $H_1(\tilde{X})$, which is finitely presentable. Any presentation matrix for $H_1(\tilde{X})$ is called an **Alexander matrix** for K. The associated ideal in Λ is the **Alexander ideal**, and, if this is principal, a generator is called the **Alexander polynomial** of K, denoted $\boldsymbol{\Delta}_K(t)$.
Finally, we state, without proof (q.v. [17]), the following theorem and an important corollary:

Theorem 3.10
If M is a Seifert matrix for a knot K, then $M - tM^{\mathrm{T}}$ is an Alexander matrix for K.

Corollary 3.11
$\Delta_K(t) = \det(M - tM^{\mathrm{T}})$, and hence $\det(K) = \Delta_K(-1)$.

3.4 The Conway potential function

In this section we examine a polynomial invariant introduced by Conway in [5]. We first consider the approach adopted by Kauffman in [13] and then discuss Conway's original formulation.

3.4.1 Kauffman's approach — the single-variable polynomial

Given a knot or link K, let M be a Seifert matrix for K.
We define the **Conway potential function**, $\boldsymbol{\Omega}_K(x) = \det(xM - x^{-1}M^{\mathrm{T}})$.
An immediate consequence of this definition, together with corollary 3.11 is:

Proposition 3.12
$\Delta_K(t^2) \doteq t\Omega_K(t)$.

Theorem 3.13

i. $\Omega_K(x)$ is an invariant of link type under ambient isotopy.

ii. $\Omega_K(x) = 1$ if K is the unknot.

iii. $\Omega_L(x) = 0$ if L is a split link.

iv. $\Omega_K(x)$ obeys the following **skein identity**:

$$\Omega\left(\begin{array}{c}\nearrow\\ \searrow\end{array}\right)(x) - \Omega\left(\begin{array}{c}\nearrow\\ \searrow\end{array}\right)(x) = (x - x^{-1})\Omega\left(\begin{array}{c}\frown\\ \smile\end{array}\right)(x)$$

where $\Omega\left(\begin{array}{c}\nearrow\\ \searrow\end{array}\right)(x), \Omega\left(\begin{array}{c}\nearrow\\ \searrow\end{array}\right)(x)$, and $\Omega\left(\begin{array}{c}\frown\\ \smile\end{array}\right)(x)$ denote the potentials of knots identical except at one crossing, where they differ as indicated.

Proof

i. This part follows from proposition 3.8 since $\Omega_K(x)$ is defined in terms of the Seifert pairing.

ii. If K is the unknot then a Seifert surface V for K may be considered to be a disc, and hence $H_1(V) \cong 0$. Thus, any Seifert matrix for V is s-equivalent to $\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$ and so

$$\begin{aligned} \Omega_K(x) &= \det\left(x\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} - x^{-1}\begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\right) \\ &= \det\begin{bmatrix} 0 & x \\ x^{-1} & 0 \end{bmatrix} \\ &= 1 \end{aligned}$$

iii. If L is split then there are disjoint oriented connected Seifert surfaces $V_1, V_2 \subset S^3$ where $L = L_1 \coprod L_2$ and $L_1 = \partial V_1, L_2 = \partial V_2$. If we excise a small disc from both V_1 and V_2 and connect the two surfaces by means of a short tube whose ends we glue to these circles,

then we have a single connected oriented surface V with boundary L. Let $\mathbf{m} \in H_1(V)$ be the homology class of a meridian on the tube. L_1 and L_2 are non-empty, so $\mathbf{m} \neq 0$. But $\theta(\mathbf{m}, \alpha) = \theta(\alpha, \mathbf{m}) = 0$ for all $\alpha \in H_1(V)$ and hence $\Omega_L(x) = \det(x\theta - x^{-1}\theta) = 0$.

iv. Let each knot or link K come equipped with a specific diagram. $\Omega_K(x)$ is then calculated from the Seifert surface for this diagram. On such a surface V, the three crossings in the skein identity are related by changing the twist on a band connecting two Seifert discs. We denote the three links by K_+, K_-, K_0 and the corresponding surfaces by V_+, V_-, V_0.

If the diagram K_+ is split then so are the diagrams K_- and K_0. Hence if either K_+ or K_- is a split diagram then the result is trivial, since by part ii split links have vanishing potential function.

It may happen that K_0 is a split diagram while K_+ and K_- are not, in which case the upper and lower strands of K_0 must be in the split halves and so K_+ and K_- are ambient isotopic, hence $\Omega_{K_+}(x) = \Omega_{K_-}(x)$, thus giving the required result.

Finally, if none of K_+, K_-, K_0 are split diagrams then $H_1(V_+) \cong H_1(V_-) \cong H_1(V_0) \oplus \mathbb{Z}$, where the extra factor of $\mathbb{Z}$ is generated by $\alpha_+ \in H_1(V_+)$ and $\alpha_- \in H_1(V_-)$, each represented by a curve passing once through the corresponding twisted band of the surface. We may extend a basis for $H_1(V_0)$ to bases for $H_1(V_+)$ and $H_1(V_-)$ we see that the Seifert matrices θ_+ and θ_- for V_+ and V_- take the form:

$$\begin{aligned} \theta_+ &= \left[\begin{array}{c|c} a+1 & \mu \\ \hline \eta & \theta_0 \end{array}\right] \\ \theta_- &= \left[\begin{array}{c|c} a & \mu \\ \hline \eta & \theta_0 \end{array}\right] \end{aligned}$$

where η is a column matrix, μ is a row matrix, θ_0 denotes a Seifert matrix for V_0 and $a = \theta_-(\alpha_-, \alpha_-) = \theta_+(\alpha_+, \alpha_+) - 1$. By substituting this into the definition of the potential function, we see:

$$\begin{aligned} \Omega_{K_+}(x) - \Omega_{K_-}(x) &= \det(x\theta_+ - x^{-1}\theta_+^{\mathrm{T}}) - \det(x\theta_- - x^{-1}\theta_-^{\mathrm{T}}) \\ &= \det\left[\begin{array}{c|c} (x-x^{-1})(a+1) & x\mu - x^{-1}\eta^{\mathrm{T}} \\ \hline x\eta - x^{-1}\mu^{\mathrm{T}} & x\theta_0 - x^{-1}\theta_0^{\mathrm{T}} \end{array}\right] \\ &\quad - \det\left[\begin{array}{c|c} (x-x^{-1})a & x\mu - x^{-1}\eta^{\mathrm{T}} \\ \hline x\eta - x^{-1}\mu^{\mathrm{T}} & x\theta_0 - x^{-1}\theta_0^{\mathrm{T}} \end{array}\right] \\ &= (x - x^{-1}) \det(x\theta_0 - x^{-1}\theta_0^{\mathrm{T}}) \\ &= (x - x^{-1})\Omega_{K_0} \end{aligned}$$

□

This result may be used as a recursive definition of $\Omega_K(x)$ for any knot or link K, but the potential function is usually presented in the form of the **Conway polynomial**:

$$\nabla_K(z) = \Omega_K(x - x^{-1})$$

In this form, the axioms become:

i. $\nabla_K(z) = 1$ if K is the unknot.

ii. $\nabla_L(z) = 0$ if L is a split link.

iii. $\nabla_{\times}(z) - \nabla_{)(}(z) = z\nabla_{\cup}(z)$

3.4.2 Conway's approach — the many-variable polynomial

Given a knot or link K, label and orient each string by the process described in section 2.3.4. Conway's potential function $\Omega_K(r, s, \ldots)$ is then defined uniquely by the following conditions:

i. $\Omega_K(r, s, \ldots) = \Omega_K(-r^{-1}, -s^{-1}, \ldots)$

ii. $\Omega_K(r, s, \ldots) = (-1)^{|K|}\Omega_K(-r, -s, \ldots)$

iii. $\Omega_{K_\pi}(r, s, \ldots) = (-1)^{|\pi|}\Omega_K(\pi(r), \pi(s), \ldots)$

iv. $\Omega_{\neg K}(r, s, \ldots) = (-1)^{|K|+1}\Omega_K(r, s, \ldots)$

The first of these conditions enables us to use the notation $\{f(r, s, \ldots)\}$ for $f(r, s, \ldots) + f(-r^{-1}, -s^{-1}, \ldots)$. In particular, $\{r\}$ stands for $r - r^{-1}$.
If the knot or link K_2 is obtained from a link K_1 by removing a string labelled r, then

$$\Omega_{K_1}(1, s, t, \ldots) = (s^{\mathrm{lk}(r,s)}t^{\mathrm{lk}(r,t)} \ldots - s^{-\mathrm{lk}(r,s)}t^{-\mathrm{lk}(r,t)} \ldots)\Omega_{K_2}(s, t, \ldots)$$

Finally, denote by $K_1 \#_r K_2$ the result of tying K_2 in an r-labelled string of K_1, then

$$\Omega_{K_1 \#_r K_2} = \{r\}\Omega_{K_1}\Omega_{K_2}$$

The skein identity Ω  $(r, s, \ldots) - \Omega$  $(r, s, \ldots) = \{r\}\Omega$  $(r, s, \ldots)$ holds, as in the single-variable case, when both strands of the crossing in question have the same label.
When considering crossings where both strands have different labels, we have the following skein identity:

$$\Omega\,\overset{r}{s}(r, s, \ldots) + \Omega\,\overset{r}{s}(r, s, \ldots) = \{rs\}\Omega\,\overset{r}{s}(r, s, \ldots)$$

Note, however, that since Conway's definition adopts a left-handed crossing convention, Kauffman's formulation of the potential function differs from Conway's in certain cases.
From condition (iv), we see that adopting the more common right-handed convention results in a change of sign for links with even numbers of components. The definitions agree, however, for proper knots. Hence, the potential function (and therefore the Alexander polynomial) does not distinguish between knots and their mirror-images.

3.5 The Kauffman bracket and the Jones polynomial

In this section we construct a polynomial invariant of regular isotopy, extend this to an invariant of ambient isotopy, and then show how this may be used as a model for the Jones polynomial (defined in [12]) using an approach described in [14].
Two diagrams are said to be **regularly isotopic** if one can be transformed into the other by means of a finite sequence of Reidemeister moves of the second and third types and their inverses.

3.5.1 The Kauffman bracket

The **Kauffman bracket** of a knot K, denoted $\langle K \rangle$ is an element of the ring $\mathbb{Z}[A, B, d]$, that is, a polynomial in A, B, d, defined by the following three axioms:

i. $\langle 0 \rangle = 1$

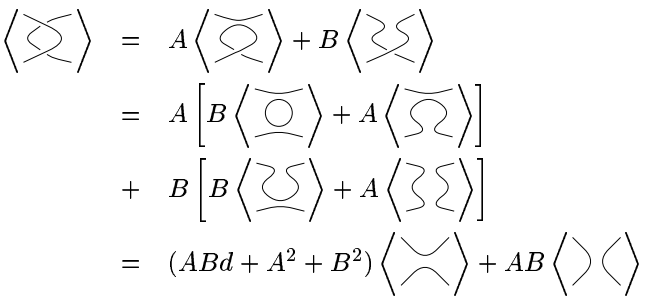
ii. $\langle 0 \cup K \rangle = d\langle K \rangle$ if K is non-empty.

iii. $\langle \times \rangle = A\langle \asymp \rangle + B\langle \rangle\langle \rangle$

This bracket polynomial is invariant under orientation-preserving homomorphisms of the plane, but not under regular or ambient isotopy. We now wish to choose appropriate values of B and d so that the bracket is an invariant of regular isotopy.

Lemma 3.14
$\langle \infty \rangle = AB\langle \rangle\langle \rangle + (ABd + A^2 + B^2)\langle \asymp \rangle$

Proof

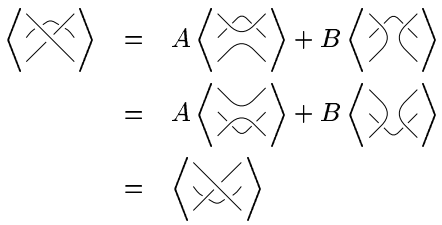


□

Corollary 3.15
The Kauffman bracket is an invariant under application of the second Reidemeister move and its inverse if $B = A^{-1}$ and $d = -A^2 - A^{-2}$.

Corollary 3.16
If $B = A^{-1}$ and $d = -A^2 - A^{-2}$ then the Kauffman bracket is an invariant of regular isotopy — in other words, invariance under R_2 and R_2^{-1} implies invariance under R_3 and R_3^{-1}

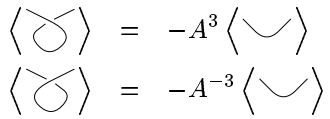
Proof



□

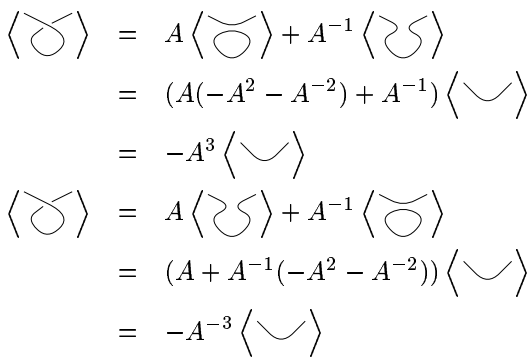
We now consider the effect of R_1 and R_1^{-1} on the bracket polynomial:

Proposition 3.17
If $B = A^{-1}$ and $d = -A^2 - A^{-2}$ then



Proof

$$\begin{aligned}
\left\langle \right\rangle &= A\left\langle \right\rangle + A^{-1}\left\langle \right\rangle \\
&= (A(-A^2 - A^{-2}) + A^{-1})\left\langle \right\rangle \\
&= -A^3\left\langle \right\rangle \\
\left\langle \right\rangle &= A\left\langle \right\rangle + A^{-1}\left\langle \right\rangle \\
&= (A + A^{-1}(-A^2 - A^{-2}))\left\langle \right\rangle \\
&= -A^{-3}\left\langle \right\rangle
\end{aligned}$$



□

3.5.2 The $\mathcal{L}$-polynomial

Thus, to extend the Kauffman bracket so that it is invariant under Reidemeister moves of the first type, we must incorporate this behaviour into a new invariant. This is done by means of the **writhe** of the knot, denoted $\mathbf{w}(K)$, which is simply the sum of the crossing numbers in the knot or link.
We define the **Kauffman $\mathcal{L}$-polynomial** to be $\mathcal{L}_K = (-A)^{-3w(K)}\langle K\rangle$.

Theorem 3.18
The polynomial $\mathcal{L}_K \in \mathbb{Z}[A, A^{-1}]$ is an invariant of ambient for oriented knots or links K.

Proof

Since $\mathcal{L}_K$ combines the behaviour of the writhe and the bracket, it is invariant under all three Reidemeister moves, and is hence an invariant of ambient isotopy. □

The power of the bracket and the $\mathcal{L}$-polynomial may be seen from the following result:

Proposition 3.19
If K^* denotes the mirror-image of the knot or link K, then

$$\begin{aligned}
\langle K^*\rangle(A) &= \langle K\rangle(A^{-1}) \\
\mathcal{L}_{K^*}(A) &= \mathcal{L}_K(A^{-1})
\end{aligned}$$

Proof

Note that switching all the crossings results in the replacement of every occurrence of A by A^{-1} in the expansion of $\langle K\rangle$. The analogous result for $\mathcal{L}_K$ follows from the fact that $w(K^*) = -w(K)$. □

The importance of the $\mathcal{L}$-polynomial lies in its ability to distinguish between a knot or link and its mirror-image; for example $\mathcal{L}_{3_1} = A^{-4} + A^{-12} - A^{-16}$, whereas $\mathcal{L}_{3_1^*} = A^4 + A^{12} - A^{16}$ (where 3_1 and 3_1^* denote the right- and left-handed trefoils). We thus obtain the following condition on amphicheiral knots and links:

Corollary 3.20
A knot or link K is amphicheiral iff $\mathcal{L}_K = \mathcal{L}_{K^*}$; that is, if $\mathcal{L}_K$ is symmetric.

This is evident in the case of the figure-of-eight knot, 4_1, since $\mathcal{L}_{4_1} = A^8 - A^4 + 1 - A^{-4} + A^{-8}$.

3.5.3 The Jones polynomial

We end this section with a brief look at the connection between $\mathcal{L}_K$ and the Jones polynomial $\mathcal{V}_K$. This powerful tool, an ambient isotopy invariant of oriented knots and links, shares with the $\mathcal{L}$-polynomial the important property of being able to distinguish between a knot or link and its mirror-image. Vaughan Jones' original paper [12] constructs the polynomial via braid theory and Von Neumann algebras and shows that it satisfies the following skein identity:

$$t^{-1}\mathcal{V}_{\times}(t) - t\mathcal{V}_{\times}(t) = \left(\sqrt{t} - \frac{1}{\sqrt{t}}\right)\mathcal{V}_{\smile}(t)$$

Together with the requirement that $\mathcal{V}_0(t) = 1$, we may calculate $\mathcal{V}_K$ for any oriented knot or link K.

Theorem 3.21
$\mathcal{V}_K(t) = \mathcal{L}_K(t^{-\frac{1}{4}})$ for any oriented knot or link K.

Proof

$$\begin{aligned} \langle \times \rangle &= A\langle \asymp \rangle + A^{-1}\langle \rangle\langle \rangle \\ \langle \times \rangle &= A^{-1}\langle \asymp \rangle + A\langle \rangle\langle \rangle \end{aligned}$$

Hence

$$A\langle \times \rangle - A^{-1}\langle \times \rangle = (A^2 - A^{-2})\langle \asymp \rangle$$

Multiplying this formula by the appropriate writhes gives:

$$A^4\mathcal{L}_{\times} - A^{-4}\mathcal{L}_{\times} = (A^{-2} - A^2)\mathcal{L}_{\smile}$$

So, by substituting $A = t^{-\frac{1}{4}}$ we get the desired result. □

A currently unsolved problem concerning $\mathcal{V}_K(t)$ is that it is as yet unknown whether $\mathcal{V}_K(t) = 1$ only if K is the unknot.
The similarities between the skein identities for the Jones and Conway polynomials led a number of mathematicians to try and generalise Jones' work. Five independent groups (P. Freyd and D. Yetter, J. Hoste, W.B.R. Lickorish and K. Millet, A. Ocneanu, and J. Prztycki and P. Tracyk) succeeded in constructing a two-variable generalised polynomial $\mathcal{P}_K$, including the Jones, Conway and Alexander polynomials as special cases [7]. This invariant, dubbed the 'HOMFLY polynomial' (Prztycki and Tracyk's paper arrived some time after the others, delayed, it was said, by the Polish postal service, and for this reason the invariant is sometimes named 'HOMFLYPT') obeys the following skein identity:

$$l\mathcal{P}_{\times}(l,m) + l^{-1}\mathcal{P}_{\times}(l,m) = m\mathcal{P}_{\smile}(l,m)$$

Chapter 4

Rational tangles, 4-plaits and two-bridge knots

Given a knot K and a diagram $D(K)$, let an **overpass** be a subarc of $D(K)$ which contains only over-crossings. Then the bridge number of $D(K)$ is the number of overcrossings in $D(K)$ and the **bridge number** of the knot K, denoted $\mathbf{b(K)}$, is the minimal bridge number over all possible diagrams for K.
Equivalently, a knot K admits an n-bridge presentation if it can be deformed so that it meets a plane $E \subset S^3$ in $2n$ points in such a way that the arcs of K contained in the half-spaces E^+ and E^- have orthogonal projections onto E which are simple and disjoint.

Lemma 4.1

i. $b(K) = 1$ iff K is the unknot.

ii. $b(K_1 \# K_2) = b(K_1) + b(K_2) - 1$.

Proposition 4.2
Two-bridge knots are prime.

Proof

Given two knots, K_1 and K_2, both nontrivial, consider their connected sum $K = K_1 \# K_2$. If K is a two-bridge knot, then from lemma 4.1 we see that $b(K) = b(K_1) + b(K_2) - 1 = 2$, so $b(K_1) + b(K_2) = 3$, implying that only one of K_1 and K_2 is nontrivial. □

4.1 Schubert's normal form

The two-bridge knots were classified by Schubert in 1956 [18] by means of their double branched covering spaces, which will be seen to be the lens spaces $L(\alpha, \beta)$.
We define the knot $S(\alpha, \beta)$ as follows: Let α and β be two coprime integers such that:

$$\alpha > 0, -\alpha < \beta < \alpha. \tag{4.1}$$

Without loss of generality we may assume that β is odd. Divide the two bridges of the knot into α segments and label them from 0 to $2\alpha - 1$ modulo 2α as shown in figure 4.1.
Counting modulo 2α, an underpass starts at the 0 end of one bridge, passes under the other bridge at β, under the first bridge at 2β, and so on. The next two results follow almost immediately upon consideration of this construction.

Proposition 4.3

i. $S(\alpha, \beta)$ is a proper knot if α is odd.

ii. $S(\alpha, \beta)$ is a two-component link if α is even.

Proof

If α is odd, then the underpass starting at the end of each bridge terminates at the other end of the other bridge, hence $S(\alpha, \beta)$ consists of one string. If α is even, the underpass starting at the end of each bridge terminates at the other end of the *same* bridge and hence $S(\alpha, \beta)$ consists of two possibly linked strings. □

Proposition 4.4

The mirror image, $S(\alpha, \beta)^*$ of $S(\alpha, \beta)$ is equivalent to $S(\alpha, -\beta)$.

Proof

This may be seen by reflecting $S(\alpha, \beta)$ in a horizontal line along the bridges. As before, when constructing the knot, we start at the 0 end of one bridge, but this time approach the other bridge from the opposite side at the crossing labelled $2\alpha - \beta \equiv -\beta \pmod{\alpha}$, which is the construction for $S(\alpha, -\beta)$. □

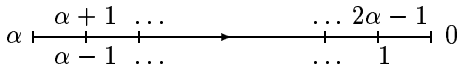


Figure 4.1: Bridge labelling

Figure 4.1 shows the knot $S(3, -1)$, a two-bridge presentation of a trefoil.

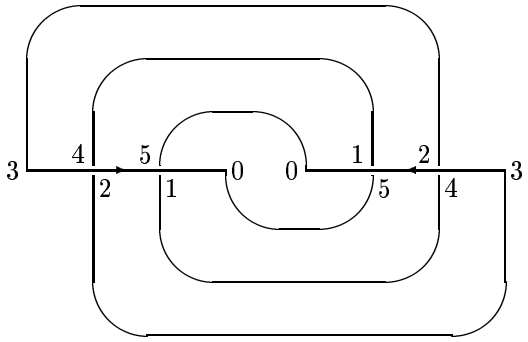


Figure 4.2: The two-bridge knot $S(3, -1)$

From the following result, we see that the 2-bridge knots cannot be completely classified by their determinants.

Proposition 4.5
The determinant of $S(\alpha, \beta)$ is α.

The **double branched cover** of S^3 over the knot K is a space C, together with a map $p : C \to S^3$ such that the preimage $p^{-1}(k)$ of any point $k \in K$ consists of a single point, and the preimage $p^{-1}(s)$ of any point $s \in S^3 \setminus K$ consists of two points.

Theorem 4.6
The double branched cover of S^3 over the knot $K = S(\alpha, \beta)$ is the lens space $L(\alpha, \beta)$

Proof
Let $B^3_- \cup B^3_+$ be a Heegaard splitting of S^3 into two 3-balls, identified along their boundaries such that $\partial B^3_- = \partial B^3_+ = S^2$, the equatorial 2-sphere of S^3. Let K be embedded in S^3 such that $B^3_- \cap K$ consists of the underpasses v_1 and v_2 of K, and $B^3_+ \cap K$ consists of the bridges w_1 and w_2 of K. Then $S^2 \cap K$ consists of four points, the endpoints of the bridges.
The double branched covers of B^3_- and B^3_+ with branch sets $v_1 \cup v_2$ and $w_1 \cup w_2$, respectively, are solid tori T_- and T_+. Each of the lifts $\tilde{w_1}$ and $\tilde{w_2}$, of the bridges of K to T_+, is a meridian of T_+ and hence each of the lifts $\tilde{v_1}$ and $\tilde{v_2}$ of v_1 and v_2 are homologous to $\alpha\mathbf{m} + \beta\mathbf{l}$ in ∂T_+ where $\mathbf{m}$ and $\mathbf{l}$ are meridian and longitude generators of $H_1(\partial T_+)$.
This is a Heegaard splitting of the lens space $L(\alpha, \beta)$. □

The classification of the two-bridge knots, then, follows from the classification of lens spaces, a proof of which appears in [3]:

Theorem 4.7 (Brody 1960)
The lens spaces $L(\alpha, \beta)$ and $L(\alpha', \beta')$ are homeomorphic iff $\alpha = \alpha'$ and $\pm\beta' \equiv \beta^{\pm 1} \pmod{\alpha}$.

From this well-known result, Schubert arrived at the following classification theorem for the two-bridge knots:

Theorem 4.8 (Schubert 1956)
i. The two-bridge links $S(\alpha, \beta)$ and $S(\alpha', \beta')$ are equivalent as unoriented links iff $\alpha = \alpha'$ and $\beta \equiv \beta' \pmod{\alpha}$.

ii. The two-bridge, two-component links $S(\alpha, \beta)$ and $S(\alpha', \beta')$ are equivalent as oriented links iff $\alpha = \alpha'$ and $\beta \equiv \beta' \pmod{2\alpha}$.

Together with Proposition 4.4 we now get the following condition for amphicheiral knots:

Corollary 4.9
$S(\alpha, \beta)$ is amphicheiral iff $\beta^2 \equiv -1 \pmod{\alpha}$.

4.2 Plaits and braids

A **braid** may be considered as a generalisation of Conway's notion of tangles, in the sense that an n-braid consists of n incoming strings and n outgoing strings, subject to the condition that the strings must run strictly downwards, as depicted in figure 4.3. Consider a rectangle (the **frame** of the braid) with n points $P_1, \ldots, P_n$ along the top and n points $Q_1, \ldots, Q_n$ along the bottom, connected by n strings $s_1, \ldots, s_n$ such that s_i joins P_i to $Q_{\pi(i)}$, where π (the **permutation** of the braid) is a permutation on $\{1, \ldots, n\}$.
A $2m$**-plait** may be formed from a $2m$-braid by adding $2m$ connecting segments as in figure 4.3.

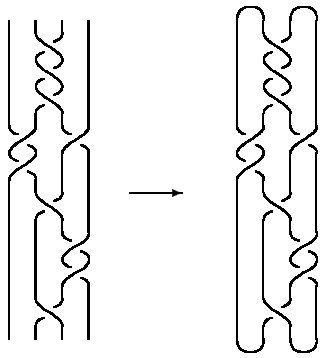


Figure 4.3: Construction of a plait from a braid

Given m-braids ζ_1, ζ_2, we can define another braid $\zeta_2\zeta_1$ by joining each emerging strand Q_i of ζ_1 to the strand P_i of ζ_2. Two braids ζ_1, ζ_2 are said to be **equivalent** if they differ by an ambient isotopy.
This operation of braid composition suggests the possibility of defining a group structure on the set of m-braids, and this is what we now do:
The m-**braid group**, denoted $\mathcal{B}_\mathbf{m}$, is the group with generators $\{\sigma_1, \ldots, \sigma_{m-1}\}$ subject to relations:

i. $\sigma_i\sigma_j = \sigma_j\sigma_i$ if $|i-j| \geqslant 2$

ii. $\sigma_{i+1}\sigma_i\sigma_{i+1} = \sigma_i\sigma_{i+1}\sigma_i$

We may regard these geometrically by considering σ_i to be the operation of interchanging strings i and $i+1$ with a right-handed crossing, as depicted in figure 4.2. The second of these relations, then, is a form of the third Reidemeister move R_3.

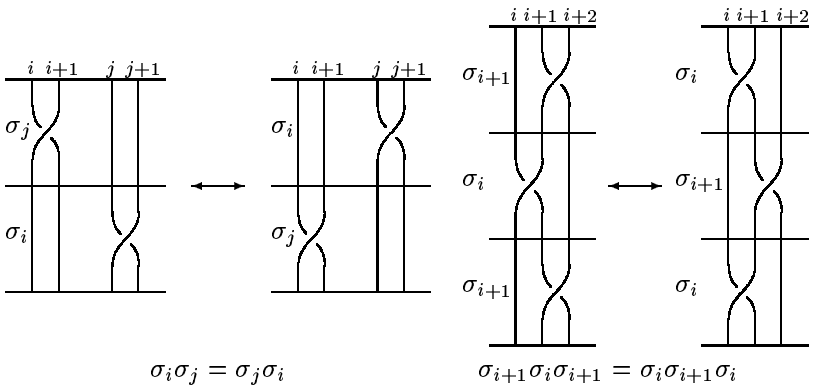


Figure 4.4: Geometric interpretation of the braid group $\mathcal{B}_m$

Theorem 4.10
The 4-plaits are exactly the 2-bridge knots.

Proof

Given a 2-bridge knot K with 4 points of intersection $P_1, \ldots, P_4$ with the projection plane $E \subset S^3$, arcs s_1, s_2 in the upper half-space E^+, and arcs t_1, t_2 in the lower half-space E^-, where s_i joins P_{2i-1} to P_{2i}, and t_i joins P_{2i} to P_{2i+1} (where $P_5 = P_1$).
Arrange the points P_i so that the orthogonal projections of the upper arcs s_i lie in a straight line. The lower arcs may then be deformed so that t_i descends strictly downwards (away from E) from P_{2i} to a single minimum, and then ascends towards E, connecting to P_{2i+1}.
The arcs s_i then become the top ends of the plait, and the minima of the arcs t_i become the bottom ends. Hence every 2-bridge knot has a 4-plait presentation.
To prove the converse observe that we may inductively move the crossings of a $2m$-plait by means of the process illustrated in figure 4.2, eventually producing a knot with m bridges at the end. □

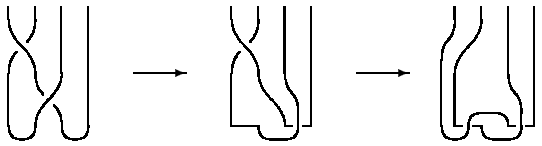


Figure 4.5: Deformation of a $2m$-plait into an m-bridge presentation

4.3 Conway's normal form

Given a tangle t, we define the **numerator** of t, denoted $\mathbf{N(t)}$ to be the knot 1^*t, and the **denominator** of t, denoted $\mathbf{D(t)}$ to be the knot 1^*t0. If t is a rational tangle, then $N(t)$ and $D(t)$ are said to be **rational knots**.

Proposition 4.11
Every rational knot has a 4-plait presentation.

Proof

Given a rational tangle t, we may straighten out the knot $N(t)$ as in figure 4.6. If t has an odd number of terms then the resulting diagram is a 4-plait. If t has an even number of terms then by a deformation of the type shown in figure 4.7 we may convert it into a 4-plait. □

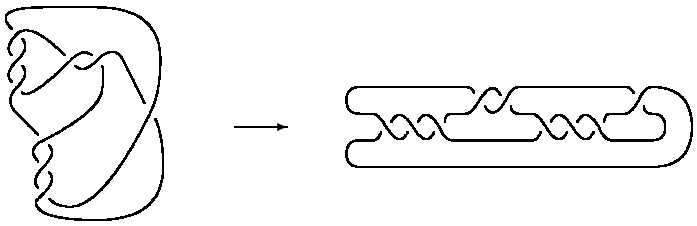


Figure 4.6: 4-plait presentation of a rational knot

We may thus think of a standard regular diagram for a rational knot as shown in figure 4.8. Such a knot or link may be denoted by $C(a_1, \ldots, a_{2k+1})$ — this is **Conway's normal form**.

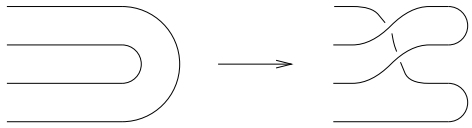


Figure 4.7: Deformation of a rational knot

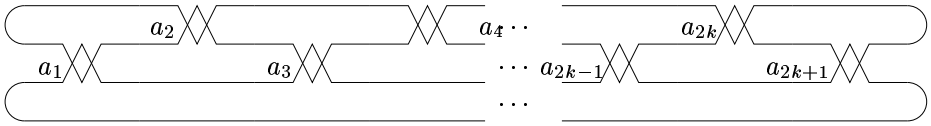


Figure 4.8: Conway's normal form

Proposition 4.12

i. $C(a_1, \ldots, a_n) \approx C(a_1, \ldots, a_n \pm 1, \mp 1)$

ii. $C(a_1, \ldots, a_n)^* \approx C(-a_1, \ldots, -a_n)$

Proof

The first part is a direct consequence of the deformation shown in figure 4.7. The second part may be seen by observing that reflecting a knot $C(a_1, \ldots, a_n)$ in the plane of the paper reverses all the crossing points, turning a right-handed twist n into a left-handed twist $-n$, hence a term a_i becomes $-a_i$. □

We now prove the converse of proposition 4.11:

Theorem 4.13

Every 4-plait (and hence every 2-bridge knot) has a presentation as a rational knot in Conway's normal form.

Proof

Given a diagram D of a 4-plait (figure 4.9), for convenience we deform it to the form shown in figure 4.10. The proof does not rely on the crossing information at the crossing points, so we shall not distinguish between under- and overcrossings.

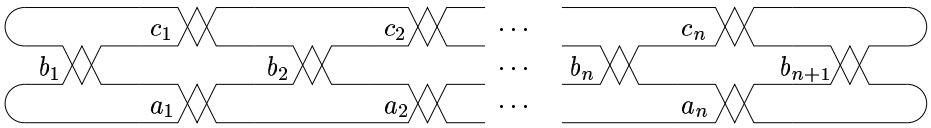


Figure 4.9: Regular diagram of a 4-plait

We wish to move all the crossing points on the left-hand side over to the right-hand side by a finite sequence of ambient isotopies. For convenience, we adopt the convention that $d_i = a_i + c_i$, and denote the diagram D in figure 4.10 by:

$$D = (a_1, \ldots, a_n | b_1, \ldots, b_n, b_{n+1} | c_1, \ldots, c_n)$$

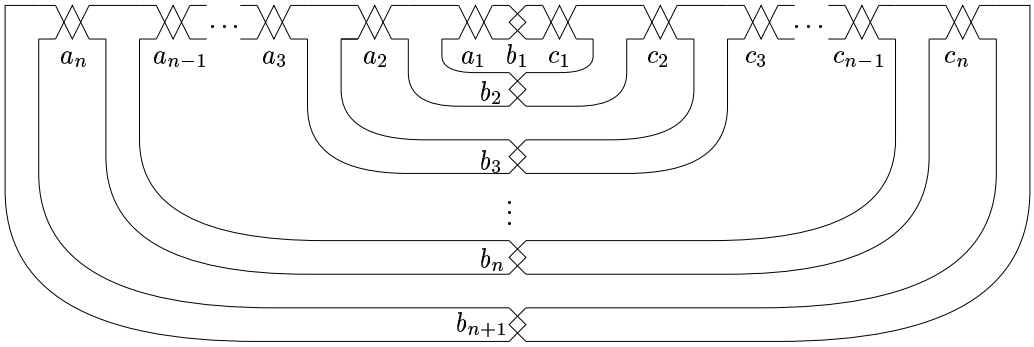


Figure 4.10: Deformed diagram of a 4-plait

Our aim, then, is to deform a diagram of this type into a diagram of the form:

$$(0, \ldots, 0 | b_1, \ldots, b_n, b_{n+1} | d_1, \ldots, d_n)$$

Firstly, we eliminate the crossing a_n by rotating the interior part of the dotted line in figure 4.11 to move the crossing over to the right-hand side. We then continue with a_{n-1}, a_{n-2} and so on.

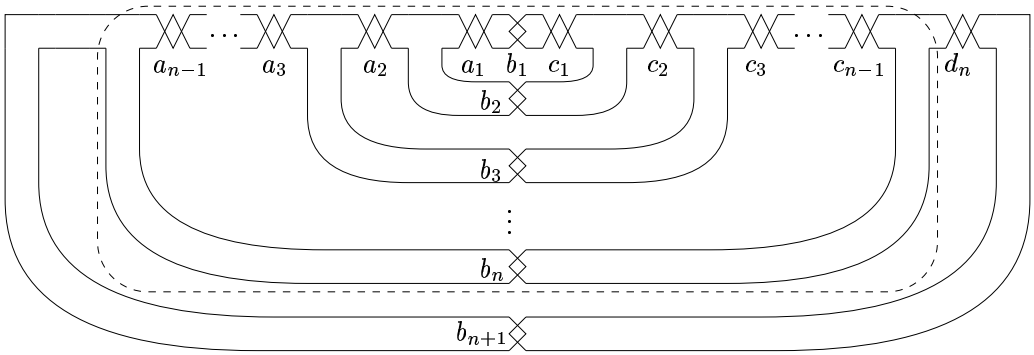


Figure 4.11: Horizontal rotation

This operation transforms a diagram

$$(a_1, \ldots, a_{i-1}, a_i, 0, \ldots, 0 | b_1, b_2, \ldots, b_n, b_{n+1} | c_1, \ldots, c_i, d_{i+1}, \ldots, d_n)$$

into a diagram

$$(a_1, \ldots, a_{i-1}, 0, \ldots, 0 | b_1, b_2, \ldots, b_n, b_{n+1} | c_1, \ldots, c_{i-1}, d_i, \ldots, d_n)$$

if a_i is even, and a diagram

$$(a_1, \ldots, a_{i-1}, 0, \ldots, 0 | b_1, \overline{b_2}, \ldots, \overline{b_i}, b_{i+1}, \ldots, b_n, b_{n+1} | c_1, \ldots, c_{i-1}, d_i, \ldots, d_n)$$

if a_i is odd, where $\overline{b_i}$ means that the i^{th} arm (with b_i crossing points) is above the horizontal axis as shown in figure 4.12

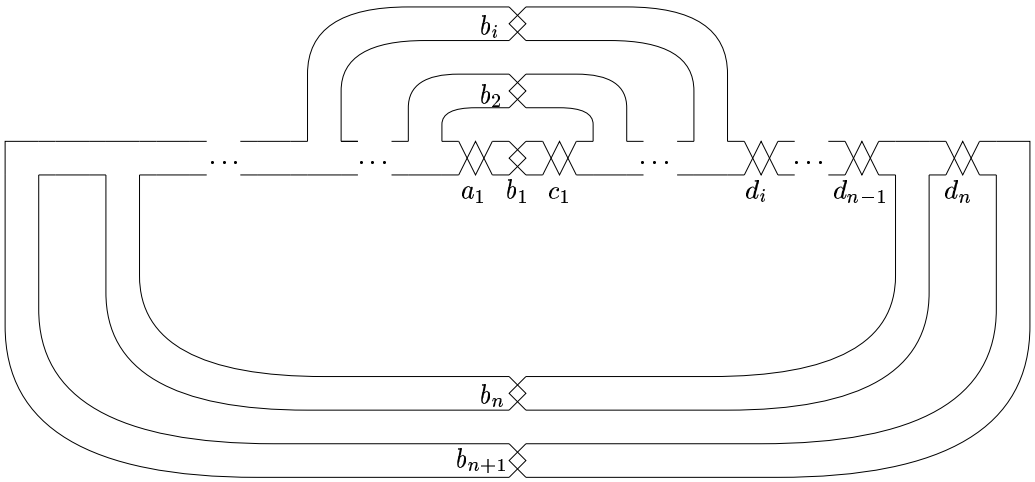


Figure 4.12: Effect of an odd horizontal rotation

To lower the arm $\overline{b_i}$, we form a new i^{th} horizontal arm with no crossing points as in figure 4.13 and rotate the interior of the dotted line b_i times around the vertical axis with the result that the b_i crossing points move to the lower arm, from the upper, which may then be removed.

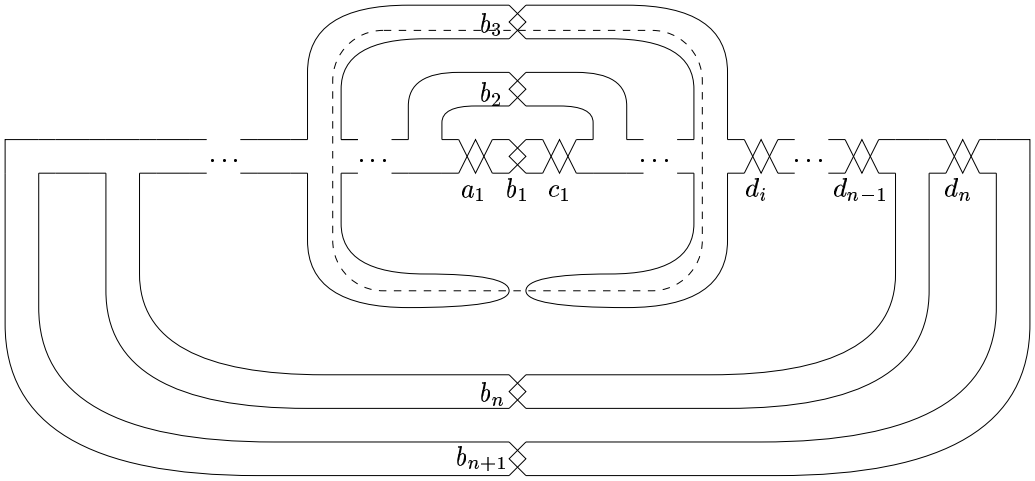


Figure 4.13: Vertical rotation

If b_i is even, this produces a diagram

$$(a_1, \ldots, a_{i-1}, 0, \ldots, 0 | b_1, \overline{b_2}, \ldots, \overline{b_i}, b_{i+1}, \ldots, b_n, b_{n+1} | c_1, \ldots, c_{i-1}, d_i, \ldots, d_n)$$

and a diagram

$$(c_1, \ldots, c_{i-1}, 0, \ldots, 0 | b_1, \overline{b_2}, \ldots, \overline{b_i}, b_{i+1}, \ldots, b_n, b_{n+1} | a_1, \ldots, a_{i-1}, d_i, \ldots, d_n)$$

if b_i is odd.

By repeated applications of such horizontal and vertical rotations, we eventually obtain a diagram

$$(a_1, 0, \ldots, 0 | b_1, \ldots, b_n, b_{n+1} | c_1, d_2, \ldots, d_n)$$

or

$$(c_1, 0, \ldots, 0 | b_1, \ldots, b_n, b_{n+1} | a_1, d_2, \ldots, d_n)$$

which may be converted to a diagram

$$(0, \ldots, 0 | b_1, \ldots, b_n, b_{n+1} | d_1, \ldots, d_n)$$

by a final horizontal rotation of the dotted area in figure 4.14

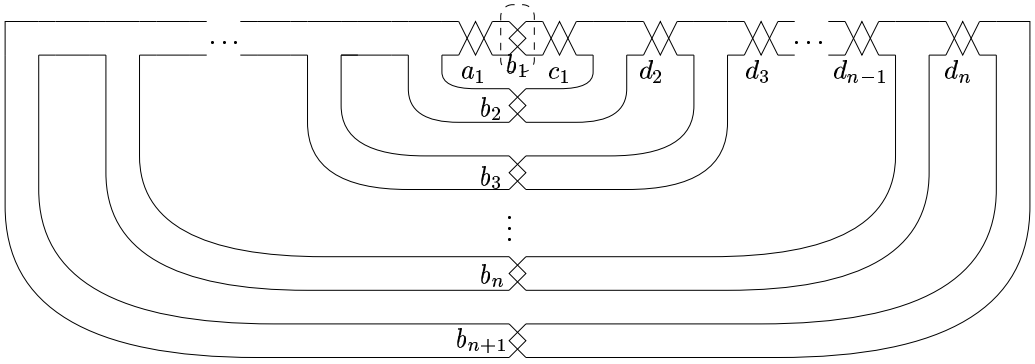


Figure 4.14: Final horizontal rotation

□

Thus, the 4-plaits (and hence the 2-bridge knots) are precisely the rational knots.
Similarly, by performing appropriate horizontal and vertical rotations on a given rational tangle, t_h, we see that:

Corollary 4.14
Given any rational tangle t, $t \doteq t_h \doteq t_v$.

It is a well-known fact from number theory that any rational number $r = \frac{p}{q}$ may be written in the form $a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \ldots}}$, where $a_1, \ldots, a_n \in \mathbb{Z}$ and n is finite. An expression of this type is called a **continued fraction** and $a_1, \ldots, a_n$ its **quotients**. For convenience we may adopt the notation $[a_1, \ldots, a_n]$ to represent such a continued fraction.
We now prove a modified version of theorem 4.6 for knots and links in Conway's normal form:

Theorem 4.15
The double branched cover of S^3 over $C(a_1, \ldots, a_n)$ is the lens space $L(\alpha, \beta)$ where $\frac{\alpha}{\beta} = [a_n, \ldots, a_1]$.

Proof

The defining braid of $K = C(a_1, \ldots, a_n)$ is $\zeta = \sigma_2^{a_n}\sigma_3^{-a_{n-1}} \ldots \sigma_2^{a_1}$. Since K is a 2-bridge knot, its double branched covering space is a lens space by theorem 4.6. The braid operations σ_2, σ_3 lift to Dehn twists δ_2, δ_3 such that

$$\begin{aligned} \delta_2(\hat{\mathbf{m}}_-) &= \hat{\mathbf{m}}_-, \quad & \delta_3(\hat{\mathbf{m}}_-) &= \hat{\mathbf{m}}_- + \hat{\mathbf{l}}_- \\ \delta_2(\hat{\mathbf{l}}_-) &= \hat{\mathbf{m}}_- - \hat{\mathbf{l}}_-, & \delta_3(\hat{\mathbf{l}}_-) &= \hat{\mathbf{l}}_- \end{aligned}$$

Thus we may assign to σ_2, σ_3 matrices:

$$\sigma_2 \mapsto A_2 = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \quad \sigma_3 \mapsto A_3 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$

describing linear mappings induced on $H_1(\partial T_-)$ by δ_2, δ_3 with respect to the basis $\{\hat{\mathbf{m}}_-, \hat{\mathbf{l}}_-\}$. Thus the braid ζ induces the transformation:

$$A = \begin{bmatrix} 1 & 0 \\ a_n & 1 \end{bmatrix} \begin{bmatrix} 1 & a_{n-1} \\ 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} 1 & a_2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ a_1 & 1 \end{bmatrix} \tag{4.2}$$

If, as in theorem 4.6, the double branched covering of this 4-plait is given by a Heegaard splitting $M = T_- \cup_h T_+$, then relative to bases $\{\hat{\mathbf{m}}_-, \hat{\mathbf{l}}_-\}$ and $\{\hat{\mathbf{m}}_+, \hat{\mathbf{l}}_+\}$ for $H_1(\partial T_-)$ and $H_1(\partial T_+)$ the isomorphism $h_* : H_1(\partial T_-) \to H_1(\partial T_+)$ is represented by the (integral, unimodular) matrix:

$$A = \begin{bmatrix} \beta & \alpha' \\ \alpha & \beta' \end{bmatrix}$$

The integers α', β' are determined up to changes $\alpha' \mapsto \alpha' + c\beta, \beta' \mapsto \beta' + c\alpha$, corresponding to a substitution $\zeta \mapsto \zeta\sigma_2^c$ which doesn't alter the plait.
The product in (4.2) defines a sequence of equations:

$$\begin{aligned} r_{n+1} &= a_n r_n + r_{n-1} \\ r_n &= a_{n-1} r_{n-1} + r_{n-2} \\ &\vdots \\ r_2 &= a_1 r_1 + 0 \\ |r_1| &= 1 \end{aligned}$$

where $r_{n+1} = \alpha, r_n = \beta$,
since:

$$\begin{bmatrix} 1 & 0 \\ -a_i & 1 \end{bmatrix} \begin{bmatrix} r_i & * \\ r_{i+1} & * \end{bmatrix} = \begin{bmatrix} r_i & * \\ r_{i+1} - a_i r_i & * \end{bmatrix} = \begin{bmatrix} r_i & * \\ r_{i-1} & * \end{bmatrix}$$

$$\begin{bmatrix} 1 & -a_{i-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} r_i & * \\ r_{i-1} & * \end{bmatrix} = \begin{bmatrix} r_i - a_{i-1} r_{i-1} & * \\ r_{i-1} & * \end{bmatrix} = \begin{bmatrix} r_{i-2} & * \\ r_{i-1} & * \end{bmatrix}$$

If we further require that $0 \leqslant r_i \leq r_{i-1}$ then this sequence of equations describes a Euclidean algorithm which is uniquely defined by the choice of α, β. Such an algorithm can be expressed by a continued fraction:

$$\frac{\alpha}{\beta} = \frac{r_{n+1}}{r_n} = a_n + \cfrac{1}{a_{n-1} + \frac{1}{a_{n-2}+\ldots}} = [a_n, a_{n-1}, \ldots, a_1] \tag{4.3}$$

□

From this result, and from the uniqueness of the continued fraction in (4.3) we derive the following elegant result:

Theorem 4.16
There exists a 1-1 correspondence between the rational knots (and hence the rational tangles) and the rational numbers; that is, $C(a_1, \ldots, a_n) \approx C(b_1, \ldots, b_m)$ iff $[a_n, \ldots, a_1] = [b_m, \ldots, b_1]$.

Corollary 4.17
2-bridge knots are alternating.

Proof

This follows from a well-known result concerning continued fractions, namely that, given a rational number $\frac{\alpha}{\beta}$, it is always possible to find a continued fraction $[a_n, \ldots, a_1] = \frac{\alpha}{\beta}$ such that the quotients a_i all have the same sign. We may thus deform $S(\alpha, \beta)$ into the form $C(a_1, \ldots, a_n)$, which, from the construction of Conway's normal form, and since $a_1, \ldots, a_n$ all have the same sign, is alternating. □

Proposition 4.18
Given two rational tangles $A = a_1 a_2 \ldots a_n$ and $B = b_1 b_2 \ldots b_m$, the knot $N(A + B)$ is a two-bridge knot and is ambient isotopic to $C(a_1, \ldots, a_{n-1}, a_n + b_m, b_{m-1}, \ldots, b_1)$.

Proof

Recall that $N(A + B)$ is constructed by joining the northeast and southeast strings of A to the northwest and southwest strings of B, and the northwest and southwest strings of A to the northeast and southeast strings of B, respectively, as in figure 4.15. Note that, by proposition 4.12, we can assume that both n and m are even.

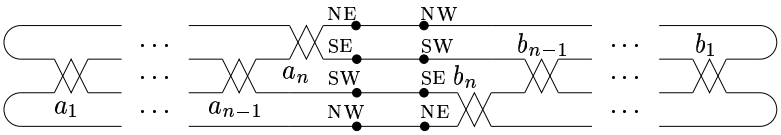


Figure 4.15: The numerator of two rational tangles

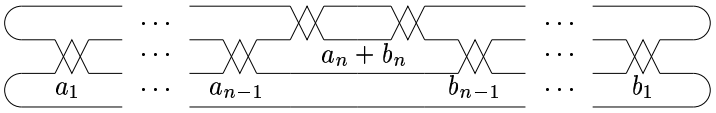


Figure 4.16: $N(A + B)$ in Conway's normal form

Then, by the process described in theorem 4.13 we may deform it into Conway's normal form to get the knot $C(a_1, \ldots, a_{n-1}, a_n + b_m, b_{m-1}, \ldots, b_1)$, as shown in figure 4.16. □

Finally, we state a result concerning the determinants of rational knots formed in this way:

Theorem 4.19
If a, b are rational tangles with fractions $\frac{p}{q}$ and $\frac{r}{s}$ then $\det N(a+b) = |ps+qr|$, and $\det N(ab) = |qs + pr|$.

Chapter 5

Slice knots and Cobordism

5.1 Slice knots

A **slice knot** is a knot K which can arise as a 3-dimensional cross-section through a (possibly knotted) 2-sphere in S^4 or $\mathbb{R}^4$. Equivalently, $K \subset S^3 = \partial D^4$ is slice if it bounds a 2-disc Δ^2 in D^4 which has a tubular neighbourhood $\Delta^2 \times D^2$ whose intersection with S^3 is a tubular neighbourhood $K \times D^2$ of K. Note that every knot in S^3 bounds a disc in D^4, the important point of the definition being the existence of the tubular neighbourhood — that is, Δ^2 is required to be locally flat.
The simplest nontrivial examples of slice knots are the stevedore's knot 6_1 and the reef knot $3_1 \# 3_1^*$, the latter shown in figure 5.1.

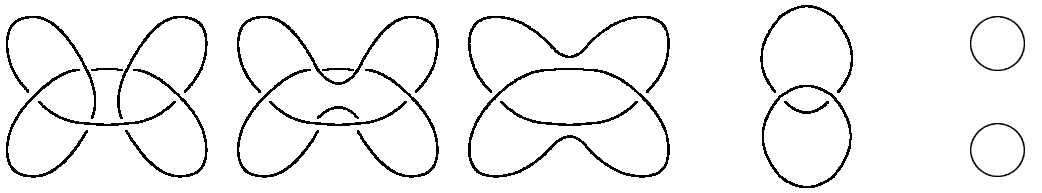


Figure 5.1: The reef knot $3_1 \# 3_1^*$ as a section through a knotted 2-sphere in S^4

We now state and prove a condition for a knot to be a slice knot — the proof follows [17], but another approach is given in [6].

Lemma 5.1
If a slice knot $K \subset S^3$ bounds a locally flat disc $\Delta \subset D^4$ and also bounds a Seifert surface $V \subset S^3$, then there exists a bicollared 3-manifold $W \subset D^4$ such that $W \cap S^3 = V$ and $\partial W = V \cup \Delta$.

Proof
(Omitted — q.v. [17].) □

Lemma 5.2
If W is a closed, connected, orientable 3-manifold such that ∂W is a connected 2-manifold

of genus g, then there exists a basis for $H_1(\partial W)$ represented by 1-cycles, half of which are 2-boundaries in W.

Proof

Consider the exact homology sequence of the pair $(W, \partial W)$:

$$\ldots \longrightarrow H_3(W,\partial W) \xrightarrow{\partial} H_2(\partial W) \longrightarrow H_2(W) \longrightarrow H_2(W,\partial W) \xrightarrow{\partial} H_1(\partial W) \longrightarrow H_1(W)$$

$$\longrightarrow H_1(W,\partial W) \xrightarrow{\partial} H_0(\partial W) \longrightarrow H_0(W) \longrightarrow H_0(W,\partial W)$$

Since $H_3(W,\partial W) \cong H_2(\partial W) \cong \mathbb{Z}$ and $H_0(\partial W) \cong H_0(W) \cong Z$, we have a short exact sequence:

$$0 \xrightarrow{h_0} H_2(W) \xrightarrow{h_1} H_2(W,\partial W) \xrightarrow{h_2} H_1(\partial W) \xrightarrow{h_3} H_1(W) \xrightarrow{h_4} H_1(W,\partial W) \xrightarrow{h_5} 0$$

Since ∂W has genus g, it follows that $H_1(\partial W)$ has rank $2g$ and so we must show that the kernel of the inclusion homomorphism $h_3 : H_1(\partial W) \longrightarrow H_1(W)$ has rank g.
Poincaré duality implies that $H_1(W,\partial W) \cong H^2(W)$ and $H_2(W,\partial W) \cong H^1(W)$. However, $\operatorname{rank} H^2(W) = \operatorname{rank} H_2(W)$ and $\operatorname{rank} H^1(W) = \operatorname{rank} H_1(W)$ and so

$$\operatorname{rank} H_2(W) = \operatorname{rank} H_1(W,\partial W) \tag{5.1}$$

$$\operatorname{rank} H_1(W) = \operatorname{rank} H_2(W,\partial W) \tag{5.2}$$

Also, for any homomorphism of abelian groups $h : G \to H$,

$$\operatorname{rank} G = \operatorname{rank}(\ker h) + \operatorname{rank}(\operatorname{im} h)$$

From this, we see that $\operatorname{rank} H_1(W,\partial W) = \operatorname{rank}\operatorname{im} h_5 + \operatorname{rank}\ker h_5 = \operatorname{rank}\ker h_5 = \operatorname{rank}\operatorname{im} h_4$, and so $\operatorname{rank}\operatorname{im} h_4 = \operatorname{rank} H_1(W,\partial W)$.
$\operatorname{rank} H_1(W) = \operatorname{rank}\operatorname{im} h_4 + \operatorname{rank}\ker h_4 = \operatorname{rank} H_1(W) + \operatorname{rank}\operatorname{im} h_3$,
so $\operatorname{rank}\operatorname{im} h_3 = \operatorname{rank} H_1(W) - \operatorname{rank} H_1(W,\partial W)$.
$\operatorname{rank} H_1(\partial W) = \operatorname{rank}\operatorname{im} h_3 + \operatorname{rank}\ker h_3$, so $\operatorname{rank}\ker h_3 = 2g - \operatorname{rank} H_1(W) + \operatorname{rank} H_1(W,\partial W)$.
Starting from the other end of the sequence, we see that $\operatorname{rank}\operatorname{im} h_0 = \operatorname{rank}\ker h_0 = 0$ and $\operatorname{rank} H_2(W) = \operatorname{rank}\operatorname{im} h_1 + \operatorname{rank}\ker h_1 = \operatorname{rank}\operatorname{im} h_1 + \operatorname{rank}\operatorname{im} h_0$, so $\operatorname{rank}\operatorname{im} h_1 = \operatorname{rank} H_2(W)$.
$\operatorname{rank} H_2(W,\partial W) = \operatorname{rank}\operatorname{im} h_2 + \operatorname{rank}\ker h_2$, so $\operatorname{rank}\ker h_3 = \operatorname{rank}\operatorname{im} h_2 = \operatorname{rank} H_2(W,\partial W) - \operatorname{rank} H_2(W)$.
By adding these two expressions for $\operatorname{rank}\ker h_3$ together, and substituting equations 5.1 and 5.2, we see that $\operatorname{rank}\ker h_3 = g$, as required. □

From these we can prove the following important condition on the Alexander polynomial of a slice knot:

Theorem 5.3
If $K \subset S^3$ is a slice knot then $\Delta_K(t) \doteq p(t)p(t^{-1})$, where $p(t)$ is a polynomial with integer coefficients.

Proof

If K bounds a locally flat disc Δ, choose $W^3 \subset D^4$ as in lemma 5.1. The inclusion homomorphism $H_1(\overset{\circ}{V}) \to H_1(V \cup \Delta) = H_1(\partial W)$ is an isomorphism and hence by lemma 5.2 there is a basis $\{a_1, \ldots, a_{2g}\}$ for $H_1(\overset{\circ}{V})$ such that $a_{g+1}, \ldots, a_{2g}$ are 2-boundaries in W. If $g+1 \leqslant i, j \leqslant 2g$ then a_i and i_+a_j bound disjoint 2-chains in D^4, so $\operatorname{lk}(a_i, i_+a_j) = 0$, and hence the Seifert matrix

for V has the form $\begin{bmatrix} B & C \\ D & 0 \end{bmatrix}$, where B, C, D are square integral matrices. From the Seifert matrix definition of the Alexander polynomial, we see:

$$\begin{aligned} \Delta_K(t) &= \det(V - tV^{\mathrm{T}}) \\ &= \det \begin{bmatrix} B - tB^{\mathrm{T}} & C - tD^{\mathrm{T}} \\ D - tC^{\mathrm{T}} & 0 \end{bmatrix} \\ &= \det(C - tD^{\mathrm{T}}) \det(D - tC^{\mathrm{T}}) \\ &= (-t)^g \det(C - tD^{\mathrm{T}}) \det(C - t^{-1}D^{\mathrm{T}}) \end{aligned}$$

Thus, the Alexander polynomial of a slice knot has the required form. □

Theorem 5.4

i. $K \# K^*$ is a slice knot for any knot K.

ii. $K_1 \# K_2$ is a slice knot if K_1, K_2 are.

iii. If K_1 and $K_1 \# K_2$ are slice knots then so is K_2.

Proof

$K \# K^*$ may be situated in S^3 so as to be symmetric with respect to a plane P as in figure 5.2. We may then spin the knot through D^4_+, the upper 4-disc of S^4 bounded by S^3, about the plane P to produce the desired locally-flat disc. The proof of parts ii and iii is omitted, but may be found in [6]. □

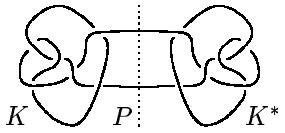


Figure 5.2: $K \# K^*$ is a slice knot

Lemma 5.5

If a nonsingular inner product on $\mathbb{Q}^{2g}$ vanishes on a subspace of dimension g, or (equivalently) has a matrix of the form:

$$\begin{bmatrix} B & C \\ C^{\mathrm{T}} & 0 \end{bmatrix}$$

with respect to some basis, where B and C are square matrices, then its signature is zero.

Proof

(Omitted — q.v. [17].) □

With this linear algebra result, we can derive the following important condition on slice knots:

Theorem 5.6

If K is a slice knot then $\sigma(K) = 0$.

Proof

Given a slice knot K with Seifert matrix M, we regard $A = M + M^{\mathrm{T}}$ as a matrix over $\mathbb{Q}$. In order to diagonalise it and determine its signature we consider it as representing a symmetric bilinear form $\langle , \rangle$ (an inner product) on $\mathbb{Q}^{2g}$ such that $\langle X, Y \rangle = XAY^{\mathrm{T}}$. This inner product is nonsingular if $\det(A) \neq 0$. A, however, is nonsingular since K is a proper knot.
By the first part of theorem 5.3, the matrix $V + V^{\mathrm{T}}$ has the form required by lemma 5.5 and hence $\sigma(K) = \sigma(V + V^{\mathrm{T}}) = 0$. □

Another interesting class of knots are the ribbon knots — K is a **ribbon knot** if it is the boundary $K = f(S^1)$ of a singular disc $f : D^2 \to S^3$ which intersects itself only in ribbon singularities, as depicted in figure 5.3.

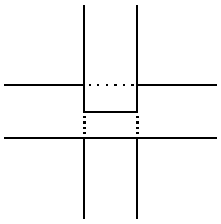


Figure 5.3: Ribbon singularity

Figure 5.4 demonstrates that the reef knot $3_1 \# 3_1^*$ is a ribbon knot, which suggests some connection between ribbon and slice knots:

Proposition 5.7
Every ribbon knot is a slice knot.

Proof

Given a ribbon knot, take a small disc around each of the singularities and push it off S^3 slightly into D^4. This removes the self-intersections, while still fulfilling the condition for the knot to be slice. □

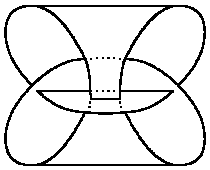


Figure 5.4: The reef knot as a ribbon knot

The converse, however, is still not known to be true in general, although no counterexample has been found yet:

Conjecture 5.8
All slice knots are ribbon knots

It is possible to generalise the concepts of ribbon and slice knots to knotted n-spheres in S^{n+2} and in the case $n \geqslant 2$ it has been shown that not every slice n-knot is a ribbon n-knot — q.v. [11].
Table 5.1 lists the slice knots with ten crossings or less (using Conway's notation) together with their Alexander polynomials.

K		$\Delta_K(t)$
0_1	∞	1
6_1	4 2	$(t^{-1}-2)(t-2)$
8_8	2 3 1 2	$(t^2-2t+2)(t^{-2}-2t^{-1}+2)$
8_9	3 1 1 3	$(t^3-2t^2+t-1)(t^{-3}-2t^{-2}+t^{-1}-1)$
8_{20}	(3, 2 1, 2–)	$(t^2-t+1)(t^{-2}-t^{-1}+1)$
9_{27}	2 1 2 1 1 2	$(t^3-2t^2+3t-1)(t^{-3}-2t^{-2}+3t^{-1}-1)$
9_{41}	2 0 : 2 0 : 2 0	$(3t^2-3t+1)(3t^{-2}-3t^{-1}+1)$
9_{46}	(3, 3, 2 1–)	$(t^{-1}-2)(t-2)$
10_3	6 4	$(3t-2)(3t^{-1}-2)$
10_{22}	3 3 1 3	$(2t^3-2t^2+2t-1)(2t^{-3}-2t^{-2}+2t-1)$
10_{35}	2 4 2 2	$(t^2-4t+2)(t^{-2}-4t^{-1}+2)$
10_{42}	2 2 1 1 1 1 2	$(t^3-3t^2+4t-1)(t^{-3}-3t^{-2}+4t^{-1}-1)$
10_{48}	(4 1, 3, 2)	$(t^4-t^3+2t^2+1)(t^{-4}-t^{-3}+2t^{-2}+1)$
10_{75}	(2 1, 2 1, 2 1+)	$(t^3-3t^2+4t-1)(t^{-3}-4t^{-2}+3t^{-1}-1)$
10_{87}	$\cdot$ 2 2 $\cdot$ 2 0	$(t-2)(t^2-t+1)(t^{-1}-2)(t^{-2}-t^{-1}+1)$
10_{99}	$\cdot$ 2 $\cdot$ 2 $\cdot$ 2 0 $\cdot$ 2 0	$(t^2-t+1)^2(t^{-2}-t^{-1}+1)^2$
10_{123}	10^*	$(t^4-3t^3+3t^2-3t+1)(t^{-4}-3t^{-3}+3t^{-2}-3t^{-1}+1)$
10_{129}	(3 2, 2 1, 2–)	$(t^2-2t+2)(t^{-2}-2t^{-1}+2)$
10_{137}	(2 2, 2 1 1, 2–)	$(t^2-3t+1)(t^{-2}-3t^{-1}+1)$
10_{140}	(4, 3, 2 1–)	$(t^2-t+1)(t^{-2}-t^{-1}+1)$
10_{153}	((3, 2)–(2 1, 2))	$(t^3-t^2+1)(t^{-3}-t^{-2}+1)$
10_{155}	$\bar{3}$: 2 : 2	$(t^3-t^2+2t-1)(t^{-3}-t^{-2}+2t^{-2}-1)$
$3_1\#3_1^*$	3 # $\bar{3}$	$(t-1+t^{-1})^2$
$4_1\#4_1^*$	2 2 # 2 2	$(t-3+t^{-1})^2$
$5_1\#5_1^*$	5 # $\bar{5}$	$(t^2-t+1-t^{-1}+t^{-2})^2$
$5_2\#5_2^*$	3 2 # $\bar{3}$ $\bar{2}$	$(2t-3+2t^{-1})^2$

Table 5.1: Slice knots with ten crossings or less

5.2 Cobordism

Define a relation $\sim$ on the class of knot types such that $K_1 \sim K_2$ iff $K_1\#K_2^*$ is a slice knot.

Proposition 5.9
The relation $\sim$ just defined is an equivalence relation.

Proof

$K\#K^*$ is slice by theorem 5.4, hence $\sim$ is reflexive. If $K_1\#K_2^*$ is a slice knot then so is $(K_1\#K_2^*)^* = K_2\#K_1^*$, and so $\sim$ is symmetric. If $K_1\#K_2^*$ and $K_2\#K_3^*$ are slice knots, then so (by theorem 5.4) is $(K_1\#K_2^*)\#(K_2\#K_3^*) = (K_1\#K_3^*)\#(K_2\#K_2^*)$, and hence (by theorem 5.4) $K_1\#K_3^*$ is slice. Thus, $\sim$ is transitive and therefore an equivalence relation. □

Theorem 5.10
$K_0 \sim K_1$ iff $\exists$ in the 4-dimensional slab of $\mathbb{R}^4$ a locally flat annulus A with boundaries K_0, K_1 such that K_0 is homologous to K_1 in A.

Proof

If such an annulus A exists then choose a vertex v below the hyperplane $\{x_4 = 0\}$ and a vertex w above the hyperplane $\{x_4 = 1\}$. The cones vK_0 and wK_1 are disjoint from each other and the interior of A, and hence the union $m = vK_0 \cup A \cup wK_1$ is then a 2-sphere with two singularities: K_0 at v and K_1^* at w.
Conversely, given a 2-sphere with just two singularities, we may deform it in such a way that it intersects the slab $\{0 \leqslant x_4 \leqslant 1\}$ in a nonsingular annulus with boundary curves the required knots. □

In view of this result we call the equivalence relation $\sim$ **cobordism**. Let G be the set of equivalence classes of knot types under cobordism, and define $[K_1]\#[K_2] = [K_1 \# K_2]$.

Proposition 5.11
The pair $(G, \#)$ forms a group (the **knot cobordism group**).

Proof

The identity element is $[0]$ (the cobordism class of the unknot), the inverse of an element $[K]$ is $[K]^* = [K^*]$. Furthermore, the operation $\#$ is associative, since $[[K_1]\#[K_2]]\#[K_3] = [K_1]\#[[K_2]\#[K_3]]$. □

Chapter 6

Machine computation

In this chapter we investigate the applications of computer techniques, in particular object-oriented programming methods, to the calculation of knot invariants.
It is far beyond the scope of this dissertation to discuss object-oriented programming techniques and design philosophy, but the interested reader may consult [19] for further details, as well as a rather solid, if labyrinthine, introduction to the C++ programming language. Those unfamiliar with the C language may wish to refer to [15].
We examine the construction of a C++ program to calculate the Kauffman bracket and $\mathcal{L}$-polynomial, the Jones polynomial, the determinant and the writhe for two-bridge knots in Conway's normal form, giving a very brief explanation of the sourcecode. Finally, we provide a table of knot invariants calculated in this way.

6.1 Preliminaries

To begin with, we briefly describe the construction of a range-checked array class, and a number of basic string-manipulation functions

6.1.1 The `vector` class

In C and C++, memory management — including use of arrays — is left entirely to the responsibility of the programmer, with any inadvertent overstepping of array bounds usually resulting in (at best) program termination with a segmentation fault or bus error, and (at worst) silent failure with an erroneous answer. It was decided fairly early on that the `polynomial` and `cnf` objects described later would need to make use of integer arrays, and so a simple user-defined class was constructed for this purpose.
The `vector` consists of an `int *` pointer, together with an `unsigned int` giving the size of the array. Constructor and destructor functions are provided, together with functionality for querying the size of, and for resizing (if required) the array. Finally, the index operator `[]` and assignment operator `=` are overloaded to behave appropriately.

6.1.2 String-manipulation functions

It was decided that the construction of a fully-specified `string` class could be easily avoided, and hence any string-handling is achieved by means of `NUL`-terminated character arrays, as is common in C.

Three such functions are provided; the first of which (`string`) makes a copy of its argument, essentially duplicating the behaviour of the standard library function `strdup`, but using the C++ `new[]` operator to allocate the required memory, rather than the more traditional C `malloc` function. The remaining two functions, `strntokens` and `strtoken` are concerned with splitting a given string into a number of tokens. The first of these returns the number of tokens in the string with the specified delimiter characters, and the second returns a copy of a particular token. Note that the memory for the strings returned by `strtoken` is dynamically allocated and should ideally be freed when no longer required, by means of the C++ `delete[]` operator.

6.2 Laurent polynomials

The `polynomial` class is intended to model Laurent polynomials and consists of an integer (representing the constant term) and two `vectors` containing the coefficients of the positive and negative powers. As well as the required constructor and destructor functions and overloaded assignment operator, a range of other interface functionality is provided. In particular, direct access to the coefficients is provided by an overloaded index operator, and the `+`, `-`, and `*` operators are overloaded to enable more intuitive addition, subtraction and multiplication of polynomials. A facility for raising a polynomial to a positive integer power is provided by the `pow` member function, and the polynomial may be displayed in TEX format by means of the `TeXformat` function, which takes an optional `char` argument, specifying the variable (with t as default).

6.3 Conway's normal form

The `cnf` class models a two-bridge knot in Conway's normal form, by encapsulating a `vector` object containing the crossing information, and an `unsigned char` describing how the braid is joined together at the ends, as shown in figure 6.1.

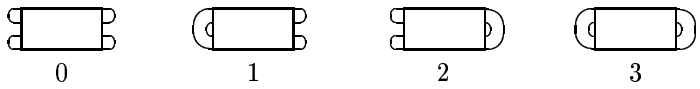


Figure 6.1: Plait ends

Constructor functions are provided to enable creation of a knot with a specified number of twists, creation of a knot from the Conway notation (making use of the string tokenising functions described earlier), together with a copy constructor and an overloaded assignment operator.
The index operator is overloaded to provide direct access to the twists, enabling examination or alteration of the knot in this way.

6.3.1 Determinant

As mentioned earlier, the determinant of a knot $C(a_1, \ldots, a_n)$ in Conway's normal form is equal to the numerator of the continued fraction $[a_n, \ldots, a_1]$. The member function `det` calculates and returns this numerator.

6.3.2 The Kauffman bracket

To calculate the Kauffman bracket we use the recursive algorithm suggested by the axioms:

i. $\langle 0 \rangle = 1$

ii. $\langle 0 \cup K \rangle = (-A^2 - A^{-2})\langle K \rangle$ if K is non-empty.

iii. $\langle \times \rangle = A\langle \asymp \rangle + A^{-1}\langle \rangle\langle \rangle$

If we represent a knot by $C(a_1, \ldots, a_n|e)$, where e describes the ends of the plait as before, then:

i. $C(0, \ldots, 0, a_i, \ldots, a_j, 0, \ldots, 0|e) \approx C(a_i, \ldots, a_j|e)$

ii. if $e = 0, 2$ then:

(a) if n is odd then

$$\langle C(a_1, \ldots, a_n|e)\rangle = \begin{cases} A\langle C(a_1+1, \ldots, a_n|e)\rangle & \\ +A^{-1}\langle C(a_1+1, \ldots, a_n|e+1)\rangle & \text{if} \quad a_1 < 0 \\ A\langle C(a_1-1, \ldots, a_n|e+1)\rangle & \\ +A^{-1}\langle C(a_1-1, \ldots, a_n|e)\rangle & \text{if} \quad a_1 > 0 \end{cases}$$

(b) if $n = 0$ then $\langle C(0|e) = \begin{cases} -A^2 - A^{-2} & \text{if} \quad e = 0 \\ 1 & \text{if} \quad e = 2 \end{cases}$

(c) if $n > 0$ and n is even,

then $\langle C(a_1, \ldots, a_n|e) = \begin{cases} (-A^3)^{-a_1}\langle C(a_2, \ldots, a_n|e)\rangle & \text{if} \quad a_1 < 0 \\ (-A^{-3})^{a_1}\langle C(a_2, \ldots, a_n|e)\rangle & \text{if} \quad a_1 > 0 \end{cases}$

iii. if $e = 1, 3$ then:

(a) if n is even and $n > 0$ then

$$\langle C(a_1, \ldots, a_n|e)\rangle = \begin{cases} A\langle C(a_1+1, \ldots, a_n|e-1)\rangle & \\ +A^{-1}\langle C(a_1+1, \ldots, a_n|e)\rangle & \text{if} \quad a_1 < 0 \\ A\langle C(a_1-1, \ldots, a_n|e)\rangle & \\ +A^{-1}\langle C(a_1-1, \ldots, a_n|e-1)\rangle & \text{if} \quad a_1 > 0 \end{cases}$$

(b) if $n = 0$ then $\langle C(0|e) = \begin{cases} 1 & \text{if} \quad e = 1 \\ -A^2 - A^{-2} & \text{if} \quad e = 3 \end{cases}$

(c) if n is odd,

then $\langle C(a_1, \ldots, a_n|e) = \begin{cases} (-A^{-3})^{-a_1}\langle C(a_2, \ldots, a_n|e)\rangle & \text{if} \quad a_1 < 0 \\ (-A^3)^{a_1}\langle C(a_2, \ldots, a_n|e)\rangle & \text{if} \quad a_1 > 0 \end{cases}$

These observations (direct consequences of the definition of the bracket polynomial and the construction of $C(a_1, \ldots, a_n|e)$) are applied recursively to calculate $\langle C(a_1, \ldots, a_n|e)\rangle$.
Table 6.1 shows the Kauffman bracket polynomial computed by the algorithm for all two-bridge knots in Conway's normal form, with less than nine crossings.

6.3.3 The writhe

The calculation of the writhe is possibly the least elegant section of the program. The algorithm implemented here essentially follows the string round the knot or link and examines the orientation at each set of crossings.

K		$\langle K \rangle$
3_1	3	$A^7 - A^3 - A^{-5}$
4_1	2 2	$A^8 - A^4 + 1 - A^{-4} + A^{-8}$
5_1	5	$A^{13} - A^9 + A^5 - A - A^{-7}$
5_2	3 2	$A^9 - A^5 + A - 2A^{-3} + A^{-7} - A^{-11}$
6_1	4 2	$A^{10} - A^6 + A^2 - 2A^{-2} + 2A^{-6} - A^{-10} + A^{-14}$
6_2	3 1 2	$A^{14} - 2A^{10} + 2A^6 - 2A^2 + 2A^{-2} - A^{-6} + A^{-10}$
6_3	2 1 1 2	$-A^{12} + 2A^8 - 2A^4 + 3 - 2A^{-4} + 2A^{-8} - A^{-12}$
7_1	7	$A^{19} - A^{15} + A^{11} - A^7 + A^3 - A^{-1} - A^{-9}$
7_2	5 2	$A^{11} - A^7 + A^3 - 2A^{-1} + 2A^{-5} - 2A^{-9} + A^{-13} - A^{-17}$
7_3	4 3	$-A^{13} + A^9 - 2A^5 + 2A - 3A^{-3} + 2A^{-7} - A^{-11} + A^{-15}$
7_4	3 1 3	$-A^{17} + 2A^{13} - 3A^9 + 2A^5 - 3A + 2A^{-3} - A^{-7} + A^{-11}$
7_5	3 2 2	$A^{15} - 2A^{11} + 3A^7 - 3A^3 + 3A^{-1} - 3A^{-5} + A^{-9} - A^{-13}$
7_6	2 2 1 2	$A^{15} - 2A^{11} + 3A^7 - 4A^3 + 3A^{-1} - 3A^{-5} + 2A^{-9} - A^{-13}$
7_7	2 1 1 1 2	$A^{15} - 3A^{11} + 3A^7 - 4A^3 + 4A^{-1} - 3A^{-5} + 2A^{-9} - A^{-13}$
8_1	6 2	$A^{12} - A^8 + A^4 - 2 + 2A^{-4} - 2A^{-8} + 2A^{-12} - A^{-16} + A^{-20}$
8_2	5 1 2	$A^{20} - 2A^{16} + 2A^{12} - 3A^8 + 3A^4 - 2 + 2A^{-4} - A^{-8} + A^{-12}$
8_3	4 4	$A^{16} - A^{12} + 2A^8 - 3A^4 + 3 - 3A^{-4} + 2A^{-8} - A^{-12} + A^{-16}$
8_4	4 1 3	$A^{20} - 2A^{16} + 3A^{12} - 3A^8 + 3A^4 - 3 + 2A^{-4} - A^{-8} + A^{-12}$
8_6	3 3 2	$A^{16} - 2A^{12} + 3A^8 - 4A^4 + 4 - 4A^{-4} + 3A^{-8} - A^{-12} + A^{-16}$
8_7	4 1 1 2	$-A^{14} + 2A^{10} - 2A^6 + 4A^2 - 4A^{-2} + 4A^{-6} - 3A^{-10} + 2A^{-14} - A^{-18}$
8_8	2 3 1 2	$-A^{18} + 2A^{14} - 3A^{10} + 5A^6 - 4A^2 + 4A^{-2} - 3A^{-6} + 2A^{-10} - A^{-14}$
8_9	3 1 1 3	$A^{16} - 2A^{12} + 3A^8 - 4A^4 + 5 - 4A^{-4} + 3A^{-8} - 2A^{-12} + A^{-16}$
8_{11}	3 2 1 2	$A^{16} - 2A^{12} + 3A^8 - 5A^4 + 5 - 4A^{-4} + 4A^{-8} - 2A^{-12} + A^{-16}$
8_{12}	2 2 2 2	$A^{16} - 2A^{12} + 4A^8 - 5A^4 + 5 - 5A^{-4} + 4A^{-8} - 2A^{-12} + A^{-16}$
8_{13}	3 1 1 1 2	$-A^{18} + 3A^{14} - 4A^{10} + 5A^6 - 5A^2 + 5A^{-2} - 3A^{-6} + 2A^{-10} - A^{-14}$
8_{14}	2 2 1 1 2	$A^{16} - 3A^{12} + 4A^8 - 5A^4 + 6 - 5A^{-4} + 4A^{-8} - 2A^{-12} + A^{-16}$

Table 6.1: Kauffman bracket for two-bridge knots with less than nine crossings

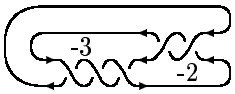


Figure 6.2: Calculation of the writhe of the knot 5_2

A value of +1 is assigned to each strand of the crossing if the direction of orientation is from left to right, and -1 otherwise, and these values are multiplied together with the number of crossings, as depicted in figure 6.2. The writhe numbers for each set of crossings are then added to calculate the total writhe of the knot.
Table 6.2 lists the determinants and writhes for all two-bridge knots of less than nine crossings.

6.3.4 The Kauffman $\mathcal{L}$-polynomial and the Jones polynomial

Having implemented the calculation of the Kauffman bracket and the writhe, together with functionality for multiplication and addition of Laurent polynomials, calculating $\mathcal{L}_K(A)$ and $\mathcal{V}_K(t)$, for a given two-bridge knot or link K, is elementary, following the procedure outlined in chapter 3.
Table 6.3 lists Jones polynomials for two-bridge knots with less than nine crossings.

K		$\det(K)$	$\mathrm{w}(K)$
3_1	3	3	-3
4_1	2 2	5	0
5_1	5	5	-5
5_2	3 2	7	-5
6_1	4 2	9	-2
6_2	3 1 2	11	-2
6_3	2 1 1 2	13	0
7_1	7	7	-7
7_2	5 2	11	-7
7_3	4 3	13	7
7_4	3 1 3	15	7
7_5	3 2 2	17	-7
7_6	2 2 1 2	19	-3
7_7	2 1 1 1 2	21	1
8_1	6 2	13	-4
8_2	5 1 2	17	-4
8_3	4 4	17	0
8_4	4 1 3	19	0
8_6	3 3 2	23	-4
8_7	4 1 1 2	23	2
8_8	2 3 1 2	25	2
8_9	3 1 1 3	25	0
8_{11}	3 2 1 2	27	-4
8_{12}	2 2 2 2	29	0
8_{13}	3 1 1 1 2	29	2
8_{14}	2 2 1 1 2	31	-4

Table 6.2: Determinant and writhe for two-bridge knots with less than nine crossings

K		$\mathcal{V}_K(t)$
3_1	3	$-t^4+t^3+t$
4_1	2 2	$t^2-t+1-t^{-1}+t^{-2}$
5_1	5	$-t^7+t^6-t^5+t^4+t^2$
5_2	3 2	$-t^6+t^5-t^4+2t^3-t^2+t$
6_1	4 2	$t^4-t^3+t^2-2t+2-t^{-1}+t^{-2}$
6_2	3 1 2	$t^5-2t^4+2t^3-2t^2+2t-1+t^{-1}$
6_3	2 1 1 2	$-t^3+2t^2-2t+3-2t^{-1}+2t^{-2}-t^{-3}$
7_1	7	$-t^{10}+t^9-t^8+t^7-t^6+t^5+t^3$
7_2	5 2	$-t^8+t^7-t^6+2t^5-2t^4+2t^3-t^2+t$
7_3	4 3	$t^{-2}-t^{-3}+2t^{-4}-2t^{-5}+3t^{-6}-2t^{-7}+t^{-8}-t^{-9}$
7_4	3 1 3	$t^{-1}-2t^{-2}+3t^{-3}-2t^{-4}+3t^{-5}-2t^{-6}+t^{-7}-t^{-8}$
7_5	3 2 2	$-t^9+2t^8-3t^7+3t^6-3t^5+3t^4-t^3+t^2$
7_6	2 2 1 2	$-t^6+2t^5-3t^4+4t^3-3t^2+3t-2+t^{-1}$
7_7	2 1 1 1 2	$-t^3+3t^2-3t+4-4t^{-1}+3t^{-2}-2t^{-3}+t^{-4}$
8_1	6 2	$t^6-t^5+t^4-2t^3+2t^2-2t+2-t^{-1}+t^{-2}$
8_2	5 1 2	$t^8-2t^7+2t^6-3t^5+3t^4-2t^3+2t^2-t+1$
8_3	4 4	$t^4-t^3+2t^2-3t+3-3t^{-1}+2t^{-2}-t^{-3}+t^{-4}$
8_4	4 1 3	$t^5-2t^4+3t^3-3t^2+3t-3+2t^{-1}-t^{-2}+t^{-3}$
8_6	3 3 2	$t^7-2t^6+3t^5-4t^4+4t^3-4t^2+3t-1+t^{-1}$
8_7	4 1 1 2	$-t^2+2t-2+4t^{-1}-4t^{-2}+4t^{-3}-3t^{-4}+2t^{-5}-t^{-6}$
8_8	2 3 1 2	$-t^3+2t^2-3t+5-4t^{-1}+4t^{-2}-3t^{-3}+2t^{-4}-t^{-5}$
8_9	3 1 1 3	$t^4-2t^3+3t^2-4t+5-4t^{-1}+3t^{-2}-2t^{-3}+t^{-4}$
8_{11}	3 2 1 2	$t^7-2t^6+3t^5-5t^4+5t^3-4t^2+4t-2+t^{-1}$
8_{12}	2 2 2 2	$t^4-2t^3+4t^2-5t+5-5t^{-1}+4t^{-2}-2t^{-3}+t^{-4}$
8_{13}	3 1 1 1 2	$-t^3+3t^2-4t+5-5t^{-1}+5t^{-2}-3t^{-3}+2t^{-4}-t^{-5}$
8_{14}	2 2 1 1 2	$t^7-3t^6+4t^5-5t^4+6t^3-5t^2+4t-2+t^{-1}$

Table 6.3: Jones polynomials for two-bridge knots with less than nine crossings

Bibliography

[1] Charilaos Aneziris.
Computer programs for knot tabulation.
q-alg, 9701006, 1997.

[2] C.W. Ashley.
The Ashley book of knots.
Faber and Faber, 1947.

[3] E. J. Brody.
The topological classification of lens spaces.
Annals of Mathematics, 71, 1960.

[4] Gerhard Burde and Heinrich Zieschang.
Knots.
de Gruyter, 1985.

[5] J. H. Conway.
An enumeration of knots and links, and some of their algebraic properties.
In John Leech, editor, *Computational Problems in Abstract Algebra*. Pergamon Press, 1970.

[6] Ralph H. Fox and John W. Milnor.
Singularities of 2-spheres in 4-space and cobordism of knots.
Osaka Journal of Mathematics, 3, 1966.

[7] Peter Freyd, David Yetter, Jim Hoste, W.B.R. Lickorish, Kenneth C. Millett, and Alexander Ocneanu.
A new polynomial invariant of knots and links.
Bulletin of the American Mathematical Society, 12(2), 1985.

[8] Eric C. Fry and Peter J. Wilson.
The Shell Combined Book of Knots and Ropework: Practical and Decorative.
David and Charles, 1981.

[9] N.D. Gilbert and T. Porter.
Knots and Surfaces.
Oxford University Press, 1994.

[10] R. Hargreaves.
Mr Clever's string book.
Price/Stern/Sloane, 1986.

[11] L.R. Hitt.
Examples of higher-dimensional slice knots which are not ribbon knots.
Proceedings of the American Mathematical Society, 77(2), 1979.

[12] Vaughan F. R. Jones.

A polynomial invariant for knots via Von Neumann algebras.
Bulletin of the American Mathematical Society, 12(1), 1985.

[13] Louis H. Kauffman.
The Conway polynomial.
Topology, 20, 1981.

[14] Louis H. Kauffman.
State models and the Jones polynomial.
Topology, 23(3), 1987.

[15] Brian W. Kernighan and Dennis M. Ritchie.
The C Programming Language.
Prentice Hall, second edition, 1988.

[16] William W. Menasco and Morwen B. Thistlethwaite.
The Tait flyping conjecture.
Bulletin of the American Mathematical Society, 25(2), October 1991.

[17] Dale Rolfsen.
Knots and Links.
Publish or Perish, 1976.

[18] H. Schubert.
Knoten mit zwei Brücken.
Mathematische Zeitschrift, 65, 1956.

[19] Bjarne Stroustrup.
The C++ Programming Language.
Addison Wesley, second edition, 1991.

[20] Peter Guthrie Tait.
On knots.
In *Scientific Papers*, volume 1. Cambridge University Press, 1898.

[21] H. F. Trotter.
Computations in knot theory.
In John Leech, editor, *Computational Problems in Abstract Algebra.* Pergamon Press, 1970.

Appendix A

C++ sourcecode

A.1 vector

A.1.1 vector.h

```
#ifndef __vector_h__
#define __vector_h__
#ifndef NULL
#define NULL 0
#endif

class vector
{
private:
  int *ia;
  unsigned int n;
public:
  vector(unsigned int = 0, const int * = NULL);
  vector(const vector&);
  ~vector(void);
  unsigned int size(void) const;
  void resize(unsigned int);
  int& operator[](unsigned int);
  int operator[](unsigned int) const;
  vector& operator=(const vector&);
};

#endif
```

A.1.2 vector.cc

```
#include <stdlib.h>
#include "vector.h"

vector::~vector(void)
{
  delete [] ia;
}
```

```
vector::vector(unsigned int sz, const int *ip) : n(sz)
{
  ia = new int [n];
  for (unsigned int i = 0; i < n; i++)
    ia[i] = (ip == NULL ? 0 : ip[i]);
}

vector::vector(const vector& v) : n(v.n)
{
  ia = new int [n];
  for (unsigned int i = 0; i < n; i++)
    ia[i] = v.ia[i];
}

unsigned int vector::size(void) const
{
  return n;
}

void vector::resize(unsigned int sz)
{
  int *ia2;
  if (n == sz)
    return;
  ia2 = new int [sz];
  for (unsigned int i = 0; i < sz; i++)
    ia2[i] = (i < n ? ia[i] : 0);
  delete [] ia;
  ia = ia2;
  n = sz;
}

int& vector::operator[](unsigned int i)
{
  if
    (i < n) return ia[i];
  else
  {
    cerr << "Attempt to reference beyond bounds of array.\n";
    exit(EXIT_FAILURE);
  }
}

int vector::operator[](unsigned int i) const
{
  if
    (i < n) return ia[i];
  else
  {
    cerr << "Attempt to reference beyond bounds of array.\n";
    exit(EXIT_FAILURE);
  }
}
```

```
vector& vector::operator=(const vector& v)
{
  delete [] ia;
  n = v.n;
  ia = new int [n];
  for (unsigned int i = 0; i < n; i++)
    ia[i] = v.ia[i];
  return *this;
}
```

A.2 String-handling functions

```
char *strtoken(const char *str, unsigned int tokn, const char *delimit)
{
  char *tstr, *rval, *rs;
  int nt = -1, l, i;
  unsigned int si = 0, intoken = 0;

  if ((str == NULL) || (strcmp(str,"") == 0) || (delimit == NULL))
    return string("");

  tstr = string(str);
  l = strlen(tstr);
  if (tokn > strntokens(str,delimit) - 1)
    return string("");
  for (i=0; i<l; i++)
  {
    if (strchr(delimit,tstr[i]) != NULL)
      intoken = 0;
    else if (!intoken)
    {
      intoken = 1;
      nt++;
    }

    if ((nt == (int) tokn) && (intoken))
    {
      si = i;
      break;
    }
  }
  rval = &tstr[si];
  for (i=si+1; i<l; i++)
    if (strchr(delimit,tstr[i]) != NULL)
    {
      tstr[i] = '\0';
      break;
    }
  rs = string(rval);
  delete [] tstr;
  return rs;
}
```

```
unsigned int strntokens(const char *str, const char *delimit)
{
  unsigned int nt = 0, intoken = 0, l;

  if (str == NULL || delimit == NULL)
    return 0;
  l = strlen(str);
  for (unsigned int i=0; i<l; i++)
    if (strchr(delimit, str[i]) != NULL)
      intoken = 0;
    else if (!intoken)
    {
      intoken = 1;
      nt++;
    }
  return nt;
}

char *string(const char *s)
{
  const char *cs, *r;
  unsigned int l;

  if (s == NULL)
    cs = "";
  else
    cs = s;

  l = strlen(cs);

  r = new char [l+1];
  strcpy(r,cs);
  return r;
}
```

A.3 polynomial

A.3.1 polynomial.h

```
#ifndef __polynomial_h__
#define __polynomial_h__

#ifndef __vector_h__
#include "vector.h"
#endif

#ifndef _IOSTREAM_H
#include <iostream.h>
#endif
```

```
class polynomial
{
private:
  vector pos;
  vector neg;
  int    con;
public:
  polynomial(int = 0, int = 0);
  polynomial(const polynomial&);

  char *TeXformat(char = 't') const;
  int pdeg(void) const;
  int ndeg(void) const;

  ostream& display(ostream&) const;

  polynomial operator+(const polynomial&) const;
  polynomial operator-(const polynomial&) const;
  polynomial operator*(const polynomial&) const;
  polynomial operator*(const int) const;
  polynomial pow(unsigned int) const;
  int operator[](int) const;
  int& operator[](int);
  polynomial& operator=(const polynomial&);
};

ostream& operator<<(ostream&, polynomial&);

#endif
```

A.3.2 polynomial.cc

```
#include <stddef.h>
#include <iostream.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "vector.h"
#include "polynomial.h"

polynomial::polynomial(int p, int n) : pos(p), neg(n), con(0)
{
}

polynomial::polynomial(const polynomial& poly)
  : pos(poly.pos), neg(poly.neg), con(poly.con)
{
}

polynomial& polynomial::operator=(const polynomial& poly)
{
  pos = poly.pos;
  neg = poly.neg;
```

```
  con = poly.con;
  return *this;
}

int polynomial::operator[](int i) const
{
  if (i > 0)
    return pos[i-1];
  else if (i < 0)
    return neg[-i-1];
  else
    return con;
}

int& polynomial::operator[](int i)
{
  if (i > 0)
    return pos[i-1];
  else if (i < 0)
    return neg[-i-1];
  else
    return con;
}

polynomial polynomial::operator+(const polynomial& p) const
{
  int tp = (pdeg() < p.pdeg() ? p.pdeg() : pdeg());
  int tn = (ndeg() < p.ndeg() ? p.ndeg() : ndeg());
  polynomial s(tp,tn);
  int i;

  for (i=0;i<tp;i++)
    s.pos[i] = (i < pdeg() ? pos[i] : 0) + (i < p.pdeg() ? p.pos[i] : 0);
  s.con = con + p.con;
  for (i=0;i<tn;i++)
    s.neg[i] = (i < ndeg() ? neg[i] : 0) + (i < p.ndeg() ? p.neg[i] : 0);
  return s;
}

polynomial polynomial::operator-(const polynomial& p) const
{
  int tp = (pdeg() < p.pdeg() ? p.pdeg() : pdeg());
  int tn = (ndeg() < p.ndeg() ? p.ndeg() : ndeg());
  polynomial s(tp,tn);
  int i;

  for (i=0;i<tp;i++)
    s.pos[i] = (i < pdeg() ? pos[i] : 0) - (i < p.pdeg() ? p.pos[i] : 0);
  s.con = con - p.con;
  for (i=0;i<tn;i++)
    s.neg[i] = (i < ndeg() ? neg[i] : 0) - (i < p.ndeg() ? p.neg[i] : 0);
  return s;
}
```

```
polynomial polynomial::operator*(const polynomial& p) const
{
  polynomial s(pdeg() + p.pdeg(), ndeg() + p.ndeg());

  for (int i = -ndeg(); i <= pdeg(); i++)
    for (int j = -p.ndeg(); j <= p.pdeg(); j++)
      s[i+j] += (*this)[i] * p[j];
  return s;
}

polynomial polynomial::operator*(const int n) const
{
  polynomial s(*this);

  for (int i=-s.ndeg(); i<=s.pdeg(); i++)
    s[i] *= n;
  return s;
}

polynomial polynomial::pow(unsigned int n) const
{
  polynomial s(0,0);

  s[0] = 1;
  for (unsigned int i=0; i<n; i++)
    s = (s * (*this));
  return s;
}

int polynomial::pdeg(void) const
{
  return pos.size();
}

int polynomial::ndeg(void) const
{
  return neg.size();
}

ostream& polynomial::display(ostream& os) const
{
  int i;

  for (i=pdeg(); i>=-ndeg(); i--)
    os << (*this)[i] << (i > -ndeg() ? " " : "");

  return os;
}

ostream& operator<<(ostream &os, polynomial &p)
{
  return p.display(os);
}
```

```
char *polynomial::TeXformat(char v) const
{
  int ft = 1;
  char tmp[4096], term[64], ssc[64], *op, *s;

  tmp[0] = '\0';

  for (int i = pdeg(); i >= -ndeg(); i--)
  {
    if ((*this)[i] == 0)
      continue;
    if (ft == 1 && (*this)[i] > 0)
      op = "";
    else
      op = ((*this)[i] > 0 ? "+" : "-");
    ft = 0;

    if (i == 1)
      ssc[0] = '\0';
    else
      sprintf(ssc,"^{%d}", i);
    if (i == 0)
      sprintf(term, "%s%d", op, abs((*this)[i]));
    else if (abs((*this)[i]) == 1)
      sprintf(term, "%s%c%s", op, v, ssc);
    else
      sprintf(term, "%s%d%c%s", op, abs((*this)[i]), v, ssc);
    sprintf(tmp,"%s%s",tmp,term);
  }

  s = new char [strlen(tmp)+1];
  sprintf(s,"%s",tmp);
  return s;
}
```

A.4 cnf

A.4.1 cnf.h

```
#ifndef __cnf_h__
#define __cnf_h__

#ifndef __vector_h__
#include "vector.h"
#endif

#ifndef __polynomial_h__
#include "polynomial.h"
#endif

#ifndef NULL
#define NULL 0
#endif
```

```
class cnf
{
private:
  vector twists;
  unsigned char   ends;
  void init(void);
  int dir(int, int) const;
  int sign(int, int, int) const;
public:
  cnf(unsigned int = 0, unsigned char = 4);
  cnf(const cnf &);
  cnf(const char *);

  unsigned int ntwists(void) const;
  int det(void) const;
  int writhe(void) const;
  polynomial bracket(void) const;
  polynomial kauffman(void) const;
  polynomial jones(void) const;
  ostream& display(ostream&) const;

  int &operator[](unsigned int);
  int operator[](unsigned int) const;

  cnf &operator=(const cnf &);
};

ostream& operator<<(ostream&, const cnf&);

#endif
```

A.4.2 cnf.cc

```
#include "vector.h"
#include "polynomial.h"
#include "cnf.h"
#include "string.h"

#include <stdlib.h>

#define ODD(a)  ((a)%2 == 1)
#define EVEN(a) ((a)%2 == 0)

static polynomial KA(1,0);
static polynomial KB(0,1);
static polynomial Kd(2,2);
static polynomial K1(0,0);
static polynomial K3a(3,0);
static polynomial K3b(0,3);
static char initK = 0;
```

```
void cnf::init(void)
{
  if (!initK)
  {
    KA[1] = 1;
    KB[-1] = 1;
    Kd[2] = -1; Kd[-2] = -1;
    K1[0] = 1;
    K3a[3] = -1; K3b[-3] = -1;
    initK = 1;
  }

cnf::cnf(unsigned int n, unsigned char e) : twists(n)
{
  init();
  if (e > 3)
    ends = (n%2 ? 0 : 1);
  else
    ends = e;
}

cnf::cnf(const cnf &c) : twists(c.twists), ends(c.ends)
{
  init();
}

cnf::cnf(const char *s) : twists(strntokens(s," \t\n"))
{
  unsigned int nt = strntokens(s," \t\n");

  init();
  ends = (twists.size()%2 ? 0 : 1);
  for (unsigned int i=0; i<nt; i++)
  {
    char *t = strtoken(s,i," \t\n");
    twists[i] = atoi(t);
    delete [] t;
  }
}

cnf &cnf::operator=(const cnf &c)
{
  twists = c.twists;
  ends = c.ends;
  return *this;
}

int &cnf::operator[](unsigned int i)
{
  return twists[i];
}
```

```
int cnf::operator[](unsigned int i) const
{
  return twists[i];
}

unsigned int cnf::ntwists(void) const
{
  return twists.size();
}

polynomial cnf::bracket(void) const
{
  cnf k(*this);

  for (unsigned int i=0; i<k.twists.size() && k.twists[i] == 0; i++)
    ;
  for (unsigned int j=k.twists.size(); j>0 && k.twists[j-1] == 0; j--)
    ;
  vector v(j < i ? 0 : j-i);
  for (unsigned int l=i;l<j;l++)
    v[l-i] = k.twists[l];
  k.twists = v;

  cnf a(k), b(k);

  switch (k.ends)
  {
    case 0:
    case 2:
      if (k.ntwists()%2)
      {
        if (a[0] < 0)
        {
          a[0] = a[0] + 1;
          b[0] = b[0] + 1;
          b.ends = b.ends + 1;
          polynomial p1(a.bracket());
          polynomial p2(b.bracket());
          return (KA * p1) + (KB * p2);
        }
        else
        {
          a[0] = a[0] - 1;
          b[0] = b[0] - 1;
          a.ends = a.ends + 1;
          polynomial p1(a.bracket());
          polynomial p2(b.bracket());
          return (KA * p1) + (KB * p2);
        }
      }
      else if (k.ntwists() == 0)
        return (k.ends == 0 ? Kd : K1);
      else
```

```
  {
    a[0] = 0;
    polynomial ab(a.bracket());
    if (k[0] < 0)
    {
      polynomial p(K3a.pow(-k[0]));
      return (p * ab);
    }
    else
    {
      polynomial p(K3b.pow(k[0]));
      return (p * ab);
    }
  }
  break;

case 1:
case 3:
  if (k.ntwists() == 0)
    return (k.ends == 3 ? Kd : K1);
  else if (k.ntwists()%2)
  {
    a[0] = 0;
    polynomial ab(a.bracket());
    if (k[0] < 0)
    {
      polynomial p(K3b.pow(-k[0]));
      return (p * ab);
    }
    else
    {
      polynomial p(K3a.pow(k[0]));
      return (p * ab);
    }
  }
  else
  {
    if (a[0] < 0)
    {
      a[0] = a[0] + 1;
      b[0] = b[0] + 1;
      a.ends = a.ends - 1;
      polynomial p1(a.bracket());
      polynomial p2(b.bracket());
      return (KA * p1) + (KB * p2);
    }
    else
    {
      a[0] = a[0] - 1;
      b[0] = b[0] - 1;
      b.ends = b.ends - 1;
      polynomial p1(a.bracket());
      polynomial p2(b.bracket());
      return (KA * p1) + (KB * p2);
```

```
        }
      }
      break;
  }
}

int cnf::det(void) const
{
  int num = 0;
  int den = 1;
  int tmp;
  unsigned int i;

  for (unsigned int i=0; i<twists.size(); i++)
  {
    num += twists[i] * den;
    tmp = num; num = den; den = tmp;
  }
  tmp = num; num = den; den = tmp;
  return num;
}

ostream& cnf::display(ostream& os) const
{
  os << "C(";
  for (unsigned int i = 0; i < twists.size() - 1; i++)
    os << twists[i] << ',';
  os << twists[twists.size() - 1] << '|' << int(ends) << ')';
  return os;
}

ostream& operator<<(ostream &os, const cnf &c)
{
  return c.display(os);
}

int cnf::writhe(void) const
{
  int w = 0;

  for (unsigned int i = 0; i<ntwists(); i++)
    if (ODD(ntwists()-i))
      w -= dir(i,1) * dir(i,2) * twists[i];
    else
      w += dir(i,0) * dir(i,1) * twists[i];
  return w;
}

polynomial cnf::kauffman(void) const
{
  polynomial kb(bracket());
  int wr(writhe());
  int p = (wr < 0 ? 3 * -wr : 0);
  int n = (wr > 0 ? 3 * wr : 0);
```

```
  polynomial w(p,n);
  if (wr < 0)
    w[p] = (p%2 ? -1 : 1);
  else
    w[-n] = (n%2 ? -1 : 1);
  return (kb * w);
}

polynomial cnf::jones(void) const
{
  polynomial kl(kauffman());
  polynomial v(kl.pdeg()/4,kl.ndeg()/4);
  int i;

  for (i = 4; i <= kl.pdeg(); i += 4)
    v[i/4] = kl[i];
  v[0] = kl[0];
  for (i = -4; i >= -kl.ndeg(); i -= 4)
    v[i/4] = kl[i];
  return v;
}

int cnf::dir(int twist, int strand) const
{
  int s1, s2, d;

  if (EVEN(det()))
  {
    s1 = (ends & 1 ? 1 : 2);
    s2 = (ends & 1 ? 0 : 1);
  }
  else
    s1 = s2 = 1;

  d = sign(s1, twist, strand);
  if (d == 0)
    d = sign(s2,twist,strand);
  return d;
}

int cnf::sign(int start, int twist, int strand) const
{
  if (twist == 0 && strand == start)
    return 1;

  for (int x=0, y = start, d = 1; !(x == twist && y == strand);)
  {
    int n = ntwists() - x;
    if (n == 0 && d == 1)
    {
      switch (y)
      {
        case 0: y = (ends & 2 ? 3 : 1); break;
```

```
        case 1: y = (ends & 2 ? 2 : 0); break;
        case 2: y = (ends & 2 ? 1 : 3); break;
        case 3: y = (ends & 2 ? 0 : 2); break;
      }
      d = -1;
    }
    else if (x == 0 && d == -1)
    {
      switch (y)
      {
        case 0: y = (ends & 1 ? 3 : 1); break;
        case 1: y = (ends & 1 ? 2 : 0); break;
        case 2: y = (ends & 1 ? 1 : 3); break;
        case 3: y = (ends & 1 ? 0 : 2); break;
      }
      d = 1;
    }
    else if ((ODD(n) && d == 1) ||
             (EVEN(n) && d == -1))
    {
      switch (y)
      {
        case 0: break;
        case 1: y = ODD(twists[ODD(n) ? x : x-1]) ? 2 : 1; break;
        case 2: y = ODD(twists[ODD(n) ? x : x-1]) ? 1 : 2; break;
        case 3: break;
      }
      x += d;
    }
    else if ((EVEN(n) && d == 1) ||
             (ODD(n) && d == -1))
    {
      switch (y)
      {
        case 0: y = ODD(twists[EVEN(n) ? x : x-1]) ? 1 : 0; break;
        case 1: y = ODD(twists[EVEN(n) ? x : x-1]) ? 0 : 1; break;
        case 2: break;
        case 3: break;
      }
      x += d;
    }
    if (x == 0 && y == start)
      return 0;
  }
  return d;
}
```