Skein based invariants and the Kauffman polynomial

Thesis submitted in accordance
with the requirements of the
University of Liverpool
for the degree of
Doctor in Philosophy

by

Nathan Derek Anthony Ryder

August 2008

Nathan Ryder

Skein based invariants and the Kauffman polynomial

Abstract

This thesis uses Kauffman skein theory to give several new results. We show a correspondence between Kauffman and Homfly satellite invariants with coefficients modulo 2, when we take certain patterns from the respective skeins of the annulus. Using stacked tangles we construct a polynomial time algorithm for calculating the Kauffman polynomial of links, and then extend the theory to give a new polynomial time algorithm for calculating the Homfly polynomial. We show that the Kauffman polynomials of genus 2 mutants can differ, and improve on existing examples showing the non-invariance of the Homfly polynomial under genus 2 mutation. By expressing twists as single crossings and smoothings in the Kauffman skein we develop an algorithm for calculating the Kauffman polynomial of pretzel links. Finally we consider the result of some calculations in the Kauffman skein of the annulus.

Acknowledgements

A thesis might be written by one person, but that one person could not possibly write it without many, many more people helping and supporting them. I'd like to take this opportunity to thank those who have helped me over the last four years.

First and foremost I have to thank my family, my mum Susan and my sisters Rebecca and Sophie, for their love, support and encouragement.

Thanks to Professor Hugh Morton, my supervisor, who has been a great mentor and a patient teacher throughout my studies.

I've made many friends while studying at Liverpool, and there are a few I want to thank in particular. I'd like to thank Shaine, Andy, John and Angela – "The Mathematicians" – thank you all for your example, your help, your humour, and your friendship; thank you Helena for being a great office-mate and for making me think more mathematically; thank you Rachel for being a fantastic and supportive friend. I'm so lucky that we did our PhDs together.

I acknowledge financial support from EPSRC and the Department of Mathematical Sciences; my thanks to both, especially for giving me the chance to attend conferences in the UK and abroad.

In my first lecture at Liverpool the lecturer began, "There is a famous proverb, 'three things come not back: the said word, the sped arrow, and the missed opportunity.'" I'd like to think that I've made the most of my time at Liverpool; thank you to everyone who has been a part of it.

This thesis is dedicated
to my mum and dad,
with love.

Contents

Introduction

The work of this thesis is centred around various results concerning the Kauffman polynomial invariant for links. The results cover a range of aspects and applications, and all are related to the Kauffman polynomial in some way. While the work of Chapter 4 is an algorithm for calculating the Homfly polynomial, it is motivated by the work of Chapter 3 related to the Kauffman polynomial.

We begin in Chapter 1 by introducing some of the background material that is necessary for the new material contained in the thesis. We begin with the preliminary notation for knots and links, discussing Reidemeister moves and presentations for links, as well as the concepts of satellite links and mutation of knots. I give the skein relations that I will take for the Kauffman and Homfly polynomial invariants throughout this thesis, except where noted otherwise.

Chapter 2 contains a proof of a recent conjecture [39] which is itself an extension of a much earlier result [54]. The result concerns a correspondence between the Kauffman and Homfly polynomials of certain satellites of links. This is proved by considering branching rules of basis elements in the Kauffman and Homfly skeins of the annulus. These are eigenvectors of meridian maps, and it is by considering them in this manner that we are able to prove the main result (Theorem 2.14):

"*Decorate each component L_i of a framed unoriented link L by $y_{\lambda(i)}$. The square of the Kauffman polynomial of this decorated link with coefficients in $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ is equal to the Homfly polynomial of L when each L_i is decorated by $Q_{\lambda(i),\lambda(i)}$ with coefficients in $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$, with the empty diagram taking the* *normalisation of 1 for both invariants.*"

In Chapter 3 we construct an algorithm for calculating the Kauffman polynomial of a link. We start with stacked k-tangles and represent them as k-sequences. We consider how braid generators act on k-sequences; the concept of *compatibility* of braid generators with k-sequences allows us to derive conditions, Propositions 3.2 and 3.3 that ensure that a k-sequence is compatible with a generator. Subsequently we show that it is possible to express an incompatible k-sequence as a linear combination of Kauffman equivalent k-sequences (Proposition 3.8). This is the foundation of an algorithm for calculating the Kauffman polynomial of a link presented as a k-plait. This algorithm works in polynomial time; while it was previously known that such a polynomial time algorithm was possible in principle [49], the algorithm presented in this thesis appears to be the first algorithm to do so.

Chapter 4 details an extension to the theory of Chapter 3, whereby we extend the construction of stacked k-tangles to oriented stacked k-tangles, allowing us to construct a polynomial time algorithm for calculating the Homfly polynomial of a link presented as a plait. While this is not the first algorithm that allows polynomial time calculation of the Homfly polynomial of a link, unlike previous algorithms it does so without needing to work from closed braid presentation of the link. We show several sets of examples whose Homfly polynomial could not be calculated using previous algorithms (owing to their braid index being too large). We end the chapter by considering extensions to the work of both chapters. Some ideas related to improving the algorithms are considered, as well as considering other situations where the principles of the algorithm could be developed.

In Chapter 5 we show the non-invariance of the Kauffman polynomial under genus 2 mutation of knots. The work of this chapter was motivated by a recent paper [15], and the results that we show in the chapter have been submitted for publication [44]. The non-invariance of the Kauffman polynomial for genus 2 mutants was assumed to be true, but was hard to show with specific examples owing to the general difficulty of calculating the Kauffman polynomial for complicated knots. We take knots presented in genus 2 handlebodies, which give us a constructive environment for developing examples. We show through an indirect method that pairs of genus 2 mutants exist which have different Kauffman polynomials: we give explicit examples, most notably those of Theorems 5.6 and 5.8. In doing so we also obtain new and more simple examples that show non-invariance of the Homfly polynomial under genus 2 mutation. We also record some interesting features about Vassilliev invariants for these examples.

Chapter 6 is an account of an algorithm for calculating the Kauffman polynomial of pretzel links. The method comes directly from considering the regular structure of pretzel links with respect to the Kauffman skein. The key result, Theorem 6.8, shows that we can take a pretzel and express its Kauffman polynomial as a linear combination of the Kauffman polynomials of much simpler diagrams. I give details of the algorithm and how it could be implemented in Maple based both on the recurrence relations that I develop and generating functions that arise from these.

In Chapter 7 I present some calculations in the Kauffman skein of the annulus which are motivated by previous results in the Homfly skein of the annulus [38]. We explore a family of examples, consisting of closed braids in the annulus with two boundary points. The results obtained are from explicit calculations for the first examples in the family, but unfortunately I was not able to realise a more general result for the family. However, I offer a conjecture (Conjecture 7.10) on the general result.

I conclude with several appendix chapters. Appendix A contains listings for the Maple implementations that I have created in relation to algorithms for calculating the Kauffman and Homfly polynomials of k-plaits. There are substantial comments for the code in both cases. In Appendix B I give plait presentations for all of the knots up to 10 crossings: while there are many resources for knots presented as braids I have not come across a list of plait presentations in all of the literature that I have seen.

Chapter 1

Background Material

1.1 Introduction

In this chapter we introduce most of the basic definitions in knot theory that we will be using within the rest of this thesis. We begin with fundamental concepts, such as what we define a knot or a link to be. Following a discussion of Reidemeister moves and framing, we look at presentations for knots in terms of braid and plait diagrams. We consider polynomial knot invariants as ways of distinguishing knots, and give definitions for the Kauffman and Homfly invariants. Finally we give the definition of mutation of knots, and the construction for creating satellites of knots.

1.2 Knots and Links

Many of the definitions given in this chapter are influenced by definitions given in [14] and [29].

Definition

A **knot**, K, is a smooth embedding of S^1 in $\mathbb{R}^3$ (or S^3). We can also consider it as a simple, closed curve without intersections in $\mathbb{R}^3$ (or S^3).

Definition

A **link**, L, of l components is an embedding of l copies of S^1 in $\mathbb{R}^3$ (or S^3); as with a knot, we can also consider it as l simple, closed curves without intersections in $\mathbb{R}^3$ (or S^3).

There will be instances when we are particularly concerned with links of more than one component, or of strictly one component; in these cases we will draw specific attention to the number of components involved. Unless otherwise stated, we will use the term knot to encompass links in general.

The unknot, in the context defined above, is a curve that is the boundary of an embedded piecewise linear disc in $\mathbb{R}^3$ (or S^3).

The fundamental problem in knot theory is being able to state whether or not two knots K_1 and K_2 are different objects, or whether K_2 is some suitably distorted version of K_1. For our purposes an initial definition of equivalence that we can give is as follows.

Definition

Knots K_1 and K_2, as defined previously, are equivalent if there is an orientation-preserving homeomorphism $f : S^3 \to S^3$ such that $f(K_1) = K_2$.

Definition

A **diagram** of a knot K is a generic projection of the curve in $\mathbb{R}^3$ to the plane with the information of how arcs cross clearly indicated, i.e., we do not mark the crossing of two arcs with a singularity, but distinguish how they cross. We allow no tangencies or intersections of three strands.

There are infinitely many possible diagrams of a knot K, depending on the projection and on the embedding of the curve. The simplest diagram of a knot is the most simple diagram of the unknot, as seen in Figure 1.1.

Definition

A knot is given an **orientation** by choosing a direction that the curve describing the knot travels. We orient a link by choosing a direction for each component of the link.

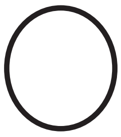


Figure 1.1: The unknot

Hence for an l component link there are 2^l ways that it can be presented as an oriented link.

For our purposes, it is convenient to consider a diagram of a knot as being equivalent to the knot itself. As we will be considering diagrams of knots we need to explore what conditions must be satisfied in order for two diagrams to be equivalent.

The diagrams in Figure 1.2 are equivalent; in the next section we consider the basic moves that allow us to relate diagrams of knots in the plane.

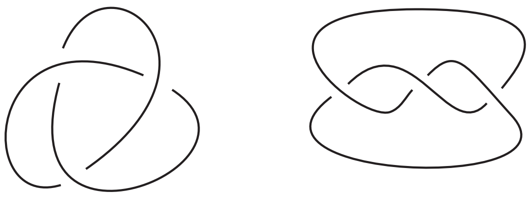


Figure 1.2: Two diagrams of the trefoil

1.2.1 Reidemeister Moves

There are three Reidemeister moves [51], which we see in Figure 1.3. These relate diagrams of knots in the plane.

The Type I move, to the left in the figure, allows us to add or remove a “kink” in the diagram. The Type II move, in the centre of the figure, shows that we can separate two arcs where one crosses over the other in two places. The Type III move, to the right of the figure, is the only one of the Reidemeister

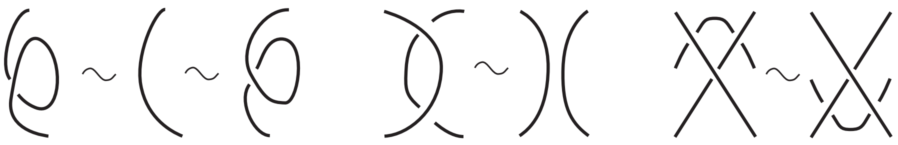


Figure 1.3: The Reidemeister Moves

moves where the number of crossings of the diagram is preserved; applications of Type I and Type II moves necessarily decrease or increase the number of crossings in the diagram.

The Reidemeister moves are essential tools as they provide the framework for deciding if two knot diagrams are equivalent.

Theorem 1.1 (Reidemeister [51]) *Two links L_1 and L_2 are equivalent if and only if a diagram of L_2 can be obtained by applying a finite number of Reidemeister moves to a diagram of L_1.*

An equivalent statement of this theorem is to say that any two diagrams of a link are related by a finite sequence of Reidemeister moves.

This is an important theorem, but at the same time it provides no insight as to how one should go about applying Reidemeister moves in order to show that two knots are equivalent.

For two different knots there will be no sequence of Reidemeister moves that takes a diagram of one to a diagram of the other, but if we do not already know that they are different objects how can we show that they are different purely by considering Reidemeister moves?

In due course we will introduce some of the properties that are used to distinguish knots. Ideally one would want a property that is easily calculable, is invariant under application of Reidemeister moves, and able to distinguish all knots; however, the tools that we possess at present do not satisfy this wish list.

1.2.2 Framed Links

Framed links are obtained by specifying a parallel curve in the neighbourhood of each component of a link; each parallel curve can be specified by an integer that is the linking number of the parallel with the original component.

A framed knot is related to a ribbon diagram by considering the knot to be described by a flat ribbon rather than a curve, with the two boundaries of the ribbon representing the original knot and its parallel. The framing of the knot is the linking number of the image of the ribbon with the knot, and we can extend this idea to consider framed links.

By drawing a link lying in the plane with the parallel running beside it we obtain the framing that is referred to as the *blackboard framing.* We can consider the blackboard framing as being obtained by converting each component to a ribbon lying flat on the plane. The Type I Reidemeister move changes the blackboard framing as it changes the number of twists in a ribbon. Type II and Type III Reidemeister moves do not change the blackboard framing.

1.3 Presentations

There are advantages to be found by considering knots and links in a particular form or format. Expressing a diagram of a knot in a certain way can sometimes be enough to distinguish it from another knot. In this section we consider two types of presentation that will be used several times in this thesis, as well as some of the consequences of their definition.

1.3.1 Braids

Artin gave the first definitions of the braid group ([3], [4]), although Gauss had previously considered braids as an interesting and useful way to record information about knotted arcs.

Geometrically we consider a word in the braid group on n strings to be n monotonically descending curves that cross over each other freely. Consider the example of Figure 1.4: this is representative of any braid in that we see no turnbacks and if we were to make a horizontal cut through the braid at any point we would meet each string only once.

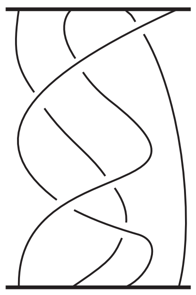


Figure 1.4: A braid on 4 strings

We denote the braid group on n strings by $\mathbb{B}_n$, and consider a generator σ_i to geometrically be the ith string crossing over the $(i+1)$th string as in Figure 1.5. We consider inverses σ_i^{-1} to be the $(i+1)$th string crossing over the ith string.

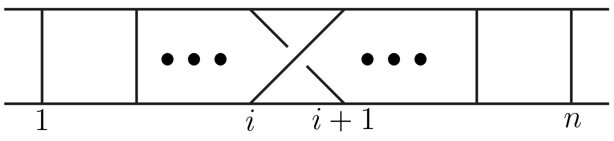


Figure 1.5: Braid generator σ_i

Thus the braid group on n strings has $n-1$ generators, $\sigma_1, \ldots, \sigma_{n-1}$, and the group has relations

$$\begin{array}{rcll} \sigma_i\sigma_j & = & \sigma_j\sigma_i & |i-j|>1 \\ \sigma_i\sigma_{i+1}\sigma_i & = & \sigma_{i+1}\sigma_i\sigma_{i+1} & 1 \leq i \leq n-2. \end{array}$$

The second relation corresponds to a Type III Reidemeister move. We close a word in the braid group by taking the endpoints at the top of the diagram to their corresponding endpoints at the bottom of the braid. The closure of a word in the braid group gives us a link (see Figure 1.6).

This leads to the following theorem.

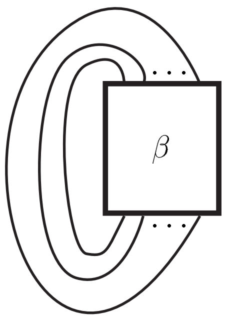


Figure 1.6: Braid closure for $\beta \in \mathbb{B}_n$

Theorem 1.2 (Alexander [1]) *Every link can be expressed as the closure of some word in the braid group $\mathbb{B}_n$ for some n.*

Definition

The **braid index** of a link, $br(L)$, is the minimum number of strings required to express it as the closure of an element in a braid group.

There are various methods for putting a diagram of a link in to a braid presentation; some of these can be difficult to implement when we consider the diagram that we start with. Also, these methods do not guarantee that the resulting word from a braid group will be a word on a minimal number of strings for the link. Expressing a link as a braid risks increasing the number of crossings in the diagram, and sometimes dramatically so ([36], [60], [61]).

Many sources state that the orientation of braid strings should be the same in a braid presentation. Orientation is important when we consider some of the invariants for knots, and will have some importance for some of the new results that we present, but we will for the most part think of braids purely in terms of how the strings lie relative to each other.

We will not consider braid presentations directly in this thesis, but we will borrow the terminology of braids for other purposes. The following format of presentation uses braid notation.

1.3.2 Plaits

The foundation of a plait presentation is the same as that for a braid presentation, namely a braid word.

Definition

A k-plait is a braid word $\beta \in \mathbb{B}_{2k}$, closed off with k caps at the top and k cups at the bottom, according to the diagram in Figure 1.7.

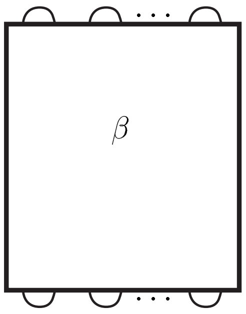


Figure 1.7: Plait presentation for $\beta \in \mathbb{B}_{2k}$

Other authors have used the term "$2k$-plat" to represent the same object we describe here; see [6] and [7] for some examples.

Theorem 1.3 *Every link has a k-plait presentation, for some k.*

Proof

Take local maxima and minima in a diagram, and drag these to the top and bottom of the picture respectively. It is possible that this will add extra crossings to the diagram due to Type II Reidemeister moves. If necessary, we comb the structure in between maxima and minima so that arcs are monotonic. ∎

At times I describe a k-plait as being a plait presentation with *width k*. As with braid presentations, plait presentations of links are not unique. The main advantage of plait presentations is that they are generally easier to obtain than braid presentations.

Briefly we need to consider bridge presentations and how they relate to plait presentations.

Definition

We can arrange a knot so that it lies completely in the plane except for a finite number of bridges – arcs whose projection to the plane result in disjoint straight lines crossing over the arcs in the plane. An embedding such as this is called a **bridge presentation**.

See Figure 1.8 for a bridge presentation of the trefoil. The original construction of bridge presentations is due to Schubert [56].

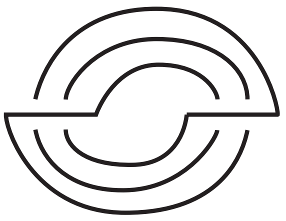


Figure 1.8: Bridge presentation of the trefoil

Definition

The **bridge number** of a bridge presentation is the number of bridges in the diagram. We define the **bridge index** as the minimum number of bridges required over all presentations for the knot.

Note that I give a slight difference in my definitions to other writers; others use the terms bridge number and bridge index to denote the same concept.

Lemma 1.4 ([9], 145 – 146) *A knot with a k-plait presentation can be presented as a diagram with bridge number k.*

This leads to a very neat result about the width of plait presentations.

Corollary 1.5 *The width of a plait presentation of a knot K is an upper bound on the bridge index of K.*

An easy example of this is the knot 6_2, which can be seen in Figure 1.9. This has an obvious 3-plait presentation, but has bridge index 2; in this case one can obtain a 2-plait presentation with little difficulty, but for more complicated knots this might not be so clear.

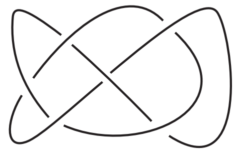


Figure 1.9: The knot 6_2

Of course, giving a plait presentation of a knot with minimal width does not guarantee that it will have minimal crossing number.

The work of Chapter 3 and Chapter 4 draws on the ideas of plait presentations in order to calculate certain knot invariants, which we now need to discuss.

1.4 Knot Invariants

The main approach that has been taken in the development of tools for distinguishing knots has been to find properties of knots, particularly properties that are invariant across all diagrams of a knot. Some of the early knot invariants and properties are relatively easy to define and obtain, but do not distinguish between many knots.

Definition

The **crossing number** of a knot, $c(K)$, is the minimal number of crossings over all diagrams of a knot.

We have already stated that showing two diagrams represent the same knot is generally a hard problem, as is showing that two diagrams are of different diagrams. Imagine a diagram of a very complicated knot; we can count the number of crossings that the diagram has, but all that this gives us is a bound on $c(K)$.

The number of knots with crossing number n grows rapidly as n increases, as we can see in Table 1.1. The knot tables are a great resource [52]; however, even if we have a diagram of a knot with minimal crossing number it may be radically different from the diagram recorded in one of the knot tables.

Crossing number	Number of knots
< 9	84
10	165
11	552
12	2176
13	9988
14	46972
15	253293

Table 1.1: Number of knots with a certain crossing number

If we are to define a property to help us distinguish between knots then ideally we need a property that is invariant across all possible diagrams for a knot K. In order for this condition to be satisfied we need a property that does not vary under application of Reidemeister moves to diagrams.

The class of invariants that we will consider in this thesis are polynomial invariants. We take a diagram of a knot and apply a method to produce a polynomial for that knot. As these properties are invariant, they do not depend on the diagram that we begin with in order to calculate the property.

Stated more formally, let p be an invariant property based on diagrams of knots; if K_1 and K_2 are diagrams of the same knot then $p(K_1) = p(K_2)$. However the converse is not always, or often, true; all of the polynomial invariants that we will discuss have examples where $p(K_1) = p(K_2)$ for diagrams K_1, K_2 that are not equivalent. A truly valuable invariant for knots would be one such that $K_1 \equiv K_2$ if and only if $p(K_1) = p(K_2)$, and where the property is readily calculable in principle: however, the complexity of a diagram might in itself impose some restriction on the ease of calculation for a property.

1.5 Polynomial Invariants

1.5.1 History

The first polynomial invariant for knots was developed by Alexander [2]. The Alexander polynomial is a property for oriented links in one variable. It cannot distinguish between reflections of knots.

Although Conway developed a polynomial invariant in the 1960s this was in fact the Alexander polynomial in another guise [12]. In the mid 1980s Jones discovered a one-variable polynomial invariant for knots that wasn't related to Alexander [22]; this was known almost immediately because it distinguished the left- and right-handed trefoils.

The Jones polynomial (for oriented links) was quickly followed by the two-variable Homfly ([17], [50]) and Kauffman ([24], [25]) polynomials.

In this thesis we are concerned with new results for the Kauffman and Homfly polynomials. We take a skein theoretic approach to calculating them, and give particular sets of skein relations for each of the invariants that we consider.

1.5.2 Homfly

There are many different ways that one can define the Homfly polynomial. There are some variations on skein relations which give the same invariant but have different algebraic properties, and we will discuss some of these as and when the need arises.

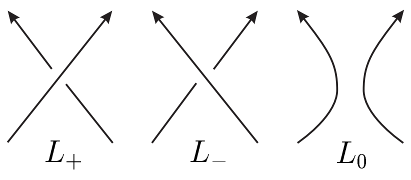


Figure 1.10: Diagrams for the Homfly skein

We consider three related diagrams, L_+, L_- and L_0, which are diagrams for oriented links that are identical except in the neighbourhood of a single crossing; in that neighbourhood we have oriented arcs as indicated in Figure 1.10.

The skein relations for the Homfly polynomial P', and for the other knot polynomials, work by relating the knot polynomials of related diagrams which differ only in the neighbourhood of a single crossing. One set of skein relations for the Homfly polynomial, in variables z and v, are

$$v^{-1}P'(L_+) - vP'(L_-) = zP'(L_0),$$

with the value of the unknot set to be 1.

For our purposes it will be convenient to use the skein relations for the framed Homfly polynomial [24]. As before, this is a polynomial in two variables z and v, and we relate the polynomials of the links L_+, L_- and L_0 with the relation

$$P(L_+) - P(L_-) = zP(L_0).$$

We set the Homfly polynomial of the regular unknot diagram to be 1, and remove a simple loop, using a Type I Reidemeister move, at the expense of multiplying by a power of $v^{\pm 1}$, according to Figure 1.11. We remove a disjoint unknot from a diagram by multiplying by $\delta = \frac{v^{-1}-v}{z}$.

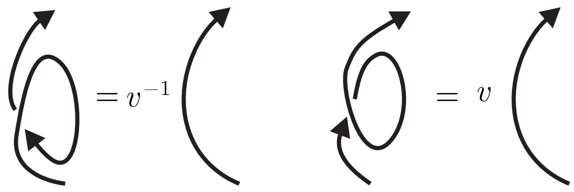


Figure 1.11: Type I Reidemeister moves in the Homfly skein

We use these framed skein relations for our calculations with plait presentations in Chapter 4. In Chapters 2, 5 and 6 we will consider taking the polynomial in terms of variables s and v, where $z = s - s^{-1}$.

If we take the diagram of a link L (that we wish to calculate the Homfly polynomial of) to be one of L_+ and L_- then we have a way of relating the Homfly polynomial of L in terms of the Homfly polynomials of two other links.

By repeating this process and removing kinks we will end up with a linear combination of unknots, which, having value 1, give us the Homfly polynomial of the original link L as the sum of the coefficients.

At the end of this section we consider problems with calculating polynomial invariants in this way; before considering the Kauffman polynomial we give some results for the Homfly polynomial that will be called on later in the thesis.

Lemma 1.6 *Reversing the orientations of all of the components of a link L leaves the Homfly polynomial invariant.*

Proof
This can be observed simply by noting that the skein relations for Homfly are unchanged by reversing the orientation of the crossings. ∎

Lemma 1.7 ([17], [28], [50]) *We can recover both the Alexander and Jones polynomials by making a substitution of variables in the Homfly polynomial. For Alexander we see that*

$$\Delta(t) = P(v = 1, z = t^{\frac{1}{2}} - t^{-\frac{1}{2}})$$

and we recover Jones with the substitution

$$V(t) = P(v = t, z = t^{\frac{1}{2}} - t^{-\frac{1}{2}}).$$

In some sense then the Homfly polynomial is a *parent invariant* of both the Alexander polynomial and the Jones polynomial.

Theorem 1.8 ([16], [35]) *Let E be the largest power of v in the Homfly polynomial of a link, and e be the smallest power of v. Then the braid index of the link, $br(L)$, is bounded in the following way:*

$$br(L) \geq \frac{1}{2}(E - e) + 1.$$

Theorem 1.8 will be of use in Chapter 4 when we look at a bound on the braid index of certain examples.

We move on to consider the polynomial invariant that we will be considering for most of this thesis.

1.5.3 Kauffman

We define the Kauffman two-variable polynomial from skein relations. This is an invariant for unoriented links, and the skein relations relate diagrams of four links. In this thesis we refer to the Dubrovnik relations for the Kauffman polynomial as in [25] and [28].

We define four links which are identical except in the neighbourhood of a single crossing; one takes a right-handed crossing (L_+, which we consider without orientation in this setting), one a left-handed crossing (L_-) and the other two take the two possible kinds of smoothing (L_0 and L_∞) as in Figure 1.12.

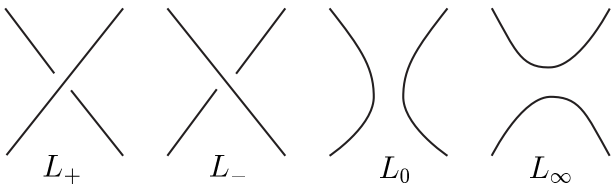


Figure 1.12: Diagrams for the Kauffman skein

The Kauffman polynomial of a link, $D(L)$ is a polynomial in two variables z and v. The value of the unknot is normalised as 1 and the main Kauffman skein relation is

$$D(L_+) - D(L_-) = z(D(L_0) - D(L_\infty)).$$

Once again we remove simple loops at the expense of multiplying by a power of $v^{\pm 1}$, according to Figure 1.13. As with the Homfly polynomial we can remove a

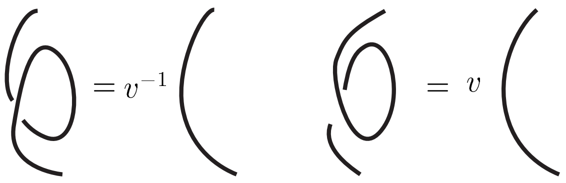


Figure 1.13: Type I Reidemeister moves in the Kauffman skein

disjoint unknot from a diagram at the expense of multiplying by $\delta = \frac{v^{-1}-v}{z} + 1$.

1.5.4 The Kauffman Skein Module

Definition

Let F be an orientable surface. The **Kauffman skein module** of $F \times I$, denoted by $K(F \times I)$, is the $\mathbb{Z}[z^{\pm 1}, v^{\pm 1}]$-module freely generated by isotopy classes of blackboard framed links in $F \times I$ including the empty link modulo the Kauffman skein relations.

In the case that F has a boundary with distinguished points, $K(F \times I)$ is the $\mathbb{Z}[z^{\pm 1}, v^{\pm 1}]$-module freely generated by isotopy classes of blackboard framed links and framed arcs connecting the distinguished points, modulo the Kauffman skein relations.

In this thesis there will be three settings that we work in. In the general setting that we have already laid out we consider $F = S^2$. In Chapter 3, when considering stacked k-tangles we will consider F as a rectangle with $2k$ points.

In Chapter 7 we will consider some calculations in the skein of the annulus, and in particular when F is the annulus with two boundary points, one point on each boundary. Elements in this skein module are composed by placing one annulus inside the other and connecting endpoints. This composition is clearly commutative.

1.5.5 Calculating Polynomial Invariants

In general, when calculating either the Homfly or Kauffman polynomial of a knot we begin by considering one diagram, and express it as a linear combination of the invariant of two or three other diagrams. We repeat the process for each of the diagrams that we have obtained, repeating again and again until we have a linear combination of disjoint unknots.

There will be situations where we can use the Type I Reidemeister move to simplify a diagram, at the expense of multiplying by a power of v, but in general we will not be able to reduce many crossings in a diagram this way. While we might be able to use the Type I move and some other tools to make the calculations easier, we are still faced with an approach that takes exponentially longer with each extra crossing that the starting diagram has.

Another approach that one might take is to use a table of invariants for knots up to a certain number of crossings, and then when our calculations reach a certain point using the previous method we can express the invariant in terms of the previously calculated invariants. However, there are over three hundred thousand knots with less than 16 crossings and this only includes objects with one component. Not only would any system working in this way need to be able to recognise which knot is being represented by a diagram, but we would also have to have a large resource that we are able to call on containing the calculated invariants.

When calculating polynomial knot invariants, even those of one variable, we reach a point where we cannot make calculations by hand. Owing to the exponential nature of the methods outlined, no matter how powerful a computer we use to aid us in our calculations we will always reach a point where we simply cannot do any more due to the number of crossings in a diagram. Perhaps this is not something that can be avoided, owing to the nature of the skein relations. However, as we shall see in Chapters 3 and 4, by restricting the setting that we work in, we can give polynomial time methods for calculating polynomial invariants of certain classes of knots.

1.6 Mutation

There are many different ways that we can define *families* of knots, i.e., knots that have some relation between them. In terms of braid diagrams, for example, we could say that the closures of braids β^m for some braid $\beta \in \mathbb{B}_n$ and $m \in \mathbb{Z}$ form an infinite family of links. We will consider some examples of this type later in the thesis.

One of the most well known concepts for a family of knots are knots that are related by mutation [12].

Definition

Consider two knots K and K'. Take a ball in S^3, T, such that K meets the boundary of T in exactly four places equally placed around the equator. Remove T and rotate it through π radians around an axis and then replace it. If by performing this action we obtain the knot K' then K and K' are said to be related by **mutation**.

If K and K' are related by mutation we say that K' is a *mutant* of K. The most well known pair of mutant knots are those of Kinoshita-Teresaka and Conway, which we see in Figure 1.14. These are the first knots in the knot

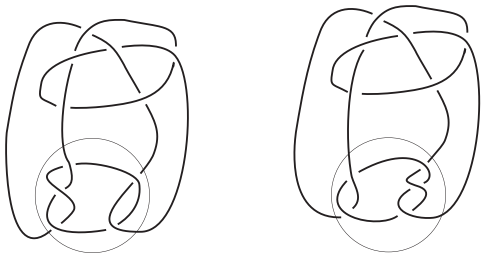


Figure 1.14: Kinoshita-Teresaka and Conway knots

table related by mutation. Mutants are an important class of knots, primarily because of the following result.

Theorem 1.9 ([28]) *Links related by mutation have identical Homfly and Kauffman polynomials. Hence they will also have the same Alexander and Jones polynomials.*

Conway first observed that the Alexander polynomial was unchanged by mutation; the observation of Lickorish is on the same principle [28]. Whichever skein relations we are using, the expression of the diagram contained in T as a linear combination of basis elements is unchanged by any of the three rotations. The contribution outside of T is unchanged, and hence K and K' will share Homfly and Kauffman polynomials.

We consider mutation in Chapter 5 in the context of genus 2 mutation and how the Homfly and Kauffman polynomials are effected by that action.

1.7 Satellites

An interesting area of study in knot theory is that of satellites of knots and links, first introduced in [55]. In Chapter 2 we consider some interesting new results regarding knot polynomials and satellites, but first we define what we mean by the satellite of a knot.

Definition

Take a framed knot K in the plane and also a framed knot P in the annulus. The knot $K * P$ is a **satellite** of K with pattern P, defined by embedding the pattern P into the neighbourhood of the curve of K.

See Figure 1.15 for an example of patterning the trefoil with a simple knotted curve from the annulus.

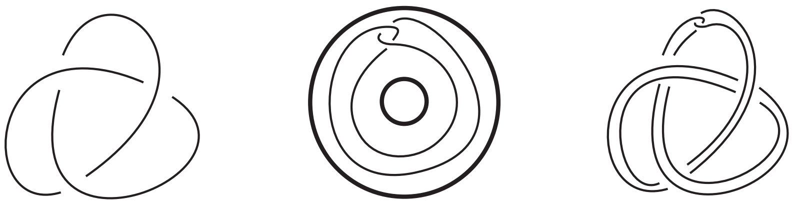


Figure 1.15: Creating a satellite of the trefoil

This is the standard way to define the satellite of a knot.

Definition

The m**-parallel** of a knot K is the satellite link obtained when the pattern P consists of the closed identity braid on m strings in the annulus.

Definition

The **reverse parallel** of a knot K is the oriented satellite link obtained when the pattern P consists of the closed identity braid on 2 strings, with the strings oriented in different directions.

See Figure 1.16 for examples of these patterns. In Chapter 2 we consider

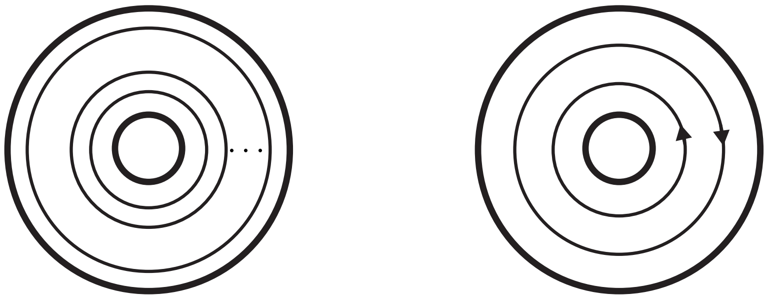


Figure 1.16: Patterns for m-parallels and reverse parallels

indexing patterns in a certain way, as linear combinations of links in the annulus. We also consider patterning a link by running different patterns around each of the components in the link.

1.7.1 Distinguishing Mutants

As well as giving interesting families of knots to consider, satellite knots also allow us to make some headway in distinguishing knots that are related by mutation. While the Homfly polynomials of two mutant knots K and K' are identical, for a suitable pattern P it can be seen that $K * P$ and $K' * P$ have different Homfly polynomials. The difference in Homfly polynomials between $K * P$ and $K' * P$ is due to the geometric difference between K and K', and so polynomial invariants of satellites can be used to distinguish knots related by mutation. Invariants of 2-parallels of knots will not distinguish mutants ([30], [48]), as the basis of the rotated tangle will not be changed by the action of the rotation, even if there are 2-parallels running through.

The rotation of the basis of these tangles will be different for m-parallels from $m = 3$ onwards [46], and there are results where certain 3-parallels distinguish mutant pairs. This gives the first opportunity for a difference in invariants, and hence a chance to distinguish mutant knots. However, there are also examples where mutant knots are not distinguished by 3-parallels and we must use more parallel curves in order to distinguish them with satellites [40].

Alexander polynomials of satellites of mutants do not differ, and so cannot be used to distinguish the knots; likewise, the Jones polynomials of cables of mutants do not differ.

As we have stated previously, many approaches to calculating polynomial invariants are exponential algorithms by nature. An undecorated m-parallel of a knot with c crossings gives a diagram with m^2c crossings. The first instance that we can use this technique of satellites to distinguish mutants is with 3-parallels, meaning that we have to consider calculating invariants of knots with $9c$ crossings. Recall that the first instance of mutant knots are the Kinoshita-Teresaka and Conway knots, each of which have 11 crossings. A 99-crossing knot is too complex for most knot polynomial algorithms that calculate from a general diagram of a knot; while there are programs and methods which have some success with satellite knots, in its general form it is a difficult problem.

Chapter 2

Homfly and Kauffman Satellite Invariants

2.1 Introduction

In this chapter we prove a conjecture of Morton on a relationship between the Kauffman polynomial of a satellite of a link and the Homfly polynomial of a reverse parallel satellite of a link [39]. This is a generalisation of a result of Rudolph which showed a certain correspondence between the Kauffman polynomial of a link and the Homfly polynomial of the reverse parallel of the link when we consider coefficients modulo 2 [54].

The background theory for the patterns for the satellites come from results in the Homfly skein of the annulus ([18], [19]) and the Kauffman skein of the annulus ([5], [31]). The patterns for the satellites are indexed by partitions, and so we begin the chapter by considering some definitions of partitions. We also show a few results (Lemmas 2.1 and 2.2) in establishing the sizes of certain sets of partitions that will be of importance in later results.

We develop the branching rules in both skeins, as these ultimately allow us to show a direct comparison between elements in the two skeins. We show by using products of meridian maps and eigenvalues that we can obtain explicit constructions for patterns in the Kauffman skein of the annulus (Lemma 2.3);

we then develop similar methods in the case of the Homfly skein of the annulus (Theorem 2.6), which we refine further when considering elements modulo 2 (Lemmas 2.11, 2.12 and 2.13).

This culminates in the proof of Conjecture 2.10 (later restated as Theorem 2.14). Throughout the chapter we develop results step-by-step so that we can then show the main result as clearly as possible.

2.1.1 Note

Throughout this chapter we consider polynomials with integer coefficients in variables v and s. We allow negative powers of these, and also denominators of products of $s^r - s^{-r}$ for $r \in \mathbb{Z} \setminus \{0\}$. It is not immediately obvious that polynomials of this type form a ring, but in Section 2.5 we show that this is the case. We denote the ring of these polynomials as $\mathbb{Z}[v^{\pm 1}, s^{\pm 1}]$.

We will also consider polynomials in variables v and s with integer coefficients modulo 2 (and with the same possible powers and denominators) which we will denote $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$. We denote the comparison between the two rings simply as “mod 2” (implicitly there is a homomorphism acting here, which we mention in Section 2.5).

2.2 Partitions

Most of the definitions of partitions were taken from the excellent introductory sections in [33].

Definition

A partition λ of a positive integer n is a sequence of natural numbers $(\lambda_1, \ldots, \lambda_k)$ with all the $\lambda_i \geq 0$ and satisfying the following conditions:

$$\begin{aligned} &\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \\ &\lambda_1 + \lambda_2 + \ldots + \lambda_k = n = |\lambda| \end{aligned}$$

A partition $\lambda = (\lambda_1, \ldots, \lambda_k)$ is said to have k parts. One way of representing a partition λ is with a Young diagram. This is a collection of n cells arranged in rows, with λ_1 cells in the first row, λ_2 cells in the second row and so on (for example, Figure 2.1).

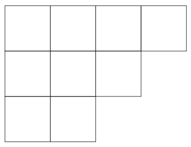


Figure 2.1: The partition $(4, 3, 2)$

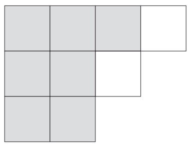


Figure 2.2: $(3, 2, 2) \subset (4, 3, 2)$

With a slight abuse of notation we denote both the partition and its Young diagram by λ. The Young diagram for $|\lambda| = 0$ is the empty diagram.

For the purposes of comparing two partitions we can add a finite number of zeros to the number sequences. A partition $\mu = (\mu_1, \ldots, \mu_k)$ is contained in a partition $\lambda = (\lambda_1, \ldots, \lambda_k)$, denoted $\mu \subset \lambda$, if $\lambda_i \geq \mu_i$, $1 \leq i \leq k$. We see this concept by considering Young diagrams for μ and λ, as in the example of Figure 2.2.

Definition

For a partition ρ define the following sets of partition:

$$\begin{aligned} \rho^+ &= \{\mu : \rho \subset \mu, |\mu| = |\rho| + 1\} \\ \rho^- &= \{\nu : \nu \subset \rho, |\nu| = |\rho| - 1\} \end{aligned}$$

Clearly $\lambda \in \rho^+ \Leftrightarrow \rho \in \lambda^-$.

Lemma 2.1 *For a partition ρ, $|\rho^+| = |\rho^-| + 1$.*

Proof

Let k be the number of distinct parts of ρ. An element of ρ^- is obtained by removing a cell from ρ, and with k distinct parts we have k cells that could be removed. Hence $|\rho^-| = k$.

An element of ρ^+ is obtained by adding a cell to ρ. ρ has k distinct parts and so there are $k+1$ locations where a cell could be added. Hence $|\rho^+| = k+1$.

Thus $|\rho^+| = |\rho^-| + 1$. ∎

Definition

For a partition ρ define the following two sets:

$$\rho^{\pm} = \left\{ \cup\ \omega^-,\ \omega \in \rho^+ \right\}\ ,\ \ \rho^{\mp} = \left\{ \cup\ \psi^+,\ \psi \in \rho^- \right\}.$$

Lemma 2.2 $\rho^{\pm} \equiv \rho^{\mp}$.

Proof

If $\omega \in \mu^-$, $\mu \in \rho^+$ and $\omega \in \nu^-$, $\nu \in \rho^+$, then either $\mu = \nu$ or $\omega = \rho$. If $\omega \neq \rho$ then its Young diagram has exactly one cell that is not in ρ, and ρ has exactly one cell that is not in ω. Thus if $\omega \in \rho^{\pm}$ then either $\omega = \rho$ or ω has exactly one cell in its Young diagram that is not in the Young diagram of ρ, and there is exactly one cell in ρ that is not in ω.

If $\psi \in \gamma^+$, $\gamma \in \rho^-$ and $\psi \in \tau^+$, $\tau \in \rho^-$, then either $\gamma = \tau$ or $\psi = \rho$. If $\psi \neq \rho$ then its Young diagram has exactly one cell that is not in ρ, and ρ has exactly one cell that is not in ψ. Then if $\psi \in \rho^{\mp}$ either $\psi = \rho$ or ψ has exactly one cell in its Young diagram that is not in the Young diagram of ρ, and there is exactly one cell in ρ that is not in ψ.

Elements in $\rho^{\pm}$ satisfy the same conditions as elements in $\rho^{\mp}$, and hence $\rho^{\pm} \equiv \rho^{\mp}$. ∎

It follows immediately from Lemma 2.2 that $\rho^{\pm} \setminus \{\rho\} \equiv \rho^{\mp} \setminus \{\rho\}$.

Definition

The **content** of a cell x in position (i, j) of the Young diagram of a partition is $c(x) = j - i$.

Content values are constant down diagonals in Young diagrams.

2.3 The Kauffman Skein of the Annulus

The initial definition of basis elements and their branching rules are due to [5], while the eigenvalues of the meridian map are due to [31].

2.3.1 Basis Elements of the Annulus

In the Kauffman skein of the annulus, $\mathcal{K}$, we have elements y_λ which are indexed by partitions λ, and form a basis of the Kauffman skein of the annulus. The y_λ are eigenvectors of the meridian map $\phi_{\mathcal{K}} : \mathcal{K} \to \mathcal{K}$ with eigenvalues

$$c_\lambda = (s - s^{-1}) \left(v^{-1} \sum_{x \in \lambda} s^{2c(x)} - v \sum_{x \in \lambda} s^{-2c(x)} \right) + \frac{v^{-1} - v}{s - s^{-1}} + 1.$$

Clearly the eigenvalues are all distinct, i.e. $c_\lambda - c_\mu = 0 \Leftrightarrow \lambda = \mu$.

The meridian map relation for $\phi_{\mathcal{K}}$ is illustrated in Figure 2.3. We consider the meridian as being placed around the annulus.

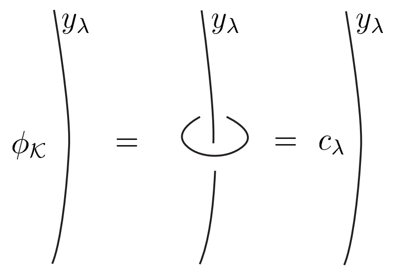


Figure 2.3: The meridian map $\phi_{\mathcal{K}}$

2.3.2 Branching Rule

The element y_1 is a single string in the skein of the annulus. Multiplication is considered as a composition of two elements in the annulus, one annulus being placed outside the other. For example, consider the composition of y_ρ and y_1 in Figure 2.4. This action is commutative.

The branching rule for the basis elements is

$$y_\rho y_1 = \sum_{\mu \in \rho^+ \cup \rho^-} y_\mu.$$

For a particular $\lambda \in \rho^+$ we can break up the branching rule to give the following expression:

$$y_\rho y_1 = y_\lambda + \sum_{\mu \in \rho^+ \cup \rho^- \setminus \{\lambda\}} y_\mu. \tag{2.1}$$

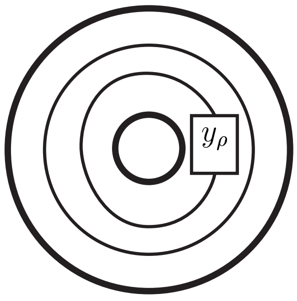


Figure 2.4: Composition of y_ρ and y_1

Definition

For partitions ρ, λ with $\lambda \in \rho^+$ define polynomial $R_\mathcal{K}(t, \rho, \lambda)$ by

$$R_\mathcal{K}(t, \rho, \lambda) = \prod_{\mu \in \rho^+ \cup \rho^- \setminus \{\lambda\}} (t - c_\mu).$$

This definition, combined with the branching rule, now allows us to give a construction for a particular element y_λ as a linear combination of meridians and longitudes based around y_ρ for $\lambda \in \rho^+$.

Lemma 2.3 *For partitions λ, ρ with $\lambda \in \rho^+$*

$$y_\lambda = \frac{R_\mathcal{K}(\phi_\mathcal{K}, \rho, \lambda)}{R_\mathcal{K}(c_\lambda, \rho, \lambda)}(y_\rho y_1).$$

Proof

Apply $R_\mathcal{K}(\phi_\mathcal{K}, \rho, \lambda)$ to both sides of the branching rule in expression 2.1. The sum in y_μ will be cancelled, as for each y_μ there will be a coefficient $\phi_\mathcal{K} - c_\mu$ which will evaluate to $c_\mu - c_\mu = 0$. Thus

$$\begin{aligned} R_\mathcal{K}(\phi_\mathcal{K}, \rho, \lambda)(y_\rho y_1) &= R_\mathcal{K}(\phi_\mathcal{K}, \rho, \lambda)\left(y_\lambda + \sum_{\mu \in \rho^+ \cup \rho^- \setminus \{\lambda\}} y_\mu\right) \\ &= R_\mathcal{K}(\phi_\mathcal{K}, \rho, \lambda)(y_\lambda) \\ \Rightarrow R_\mathcal{K}(\phi_\mathcal{K}, \rho, \lambda)(y_\rho y_1) &= R_\mathcal{K}(c_\lambda, \rho, \lambda) y_\lambda \end{aligned}$$

since $(\phi_\mathcal{K} - c_\mu)(y_\lambda) = (c_\lambda - c_\mu)y_\lambda$ by definition.

Eigenvalues c_λ are all distinct, and so $R_{\mathcal{K}}(c_\lambda, \rho, \lambda) \in \mathbb{Z}[v^{\pm 1}, s^{\pm 1}]$ is non-zero. Hence we can divide both sides of the expression by $R_{\mathcal{K}}(c_\lambda, \rho, \lambda)$, giving the result required. ∎

Note y_ρ can be expressed as a linear combination of some y_τ (for some $\tau \subset \rho, |\rho| = |\tau|+1$). Thus any y_λ can be expressed as a linear combination of linked up longitudes and meridians with coefficients from the Kauffman skein of the annulus.

We will later consider coefficients modulo 2, and we need to show that certain eigenvalues are distinct modulo 2. We show a more general result and then show the required result by corollary.

Theorem 2.4 *For partitions λ and μ,*

$$c_\lambda - c_\mu \equiv 0 \ mod\ n \Leftrightarrow \lambda = \mu, \ \ n \in \mathbb{N}, \ \ n \geq 2.$$

Proof

Clearly $\lambda = \mu \Rightarrow c_\lambda - c_\mu \equiv 0$ mod n.

Take two partitions $\lambda = (\lambda_1, \ldots, \lambda_k), \mu = (\mu_1, \ldots, \mu_l)$ such that $c_\lambda - c_\mu \equiv 0$ mod n. Let z_λ^* be the cell in position $(k, 1)$ in the Young diagram of λ (see Figure 2.5) and z_μ^* be the cell in position $(l, 1)$ in the Young diagram of μ. Content values proceed along diagonals in Young diagrams so $c(z_\lambda^*)$ and $c(z_\mu^*)$

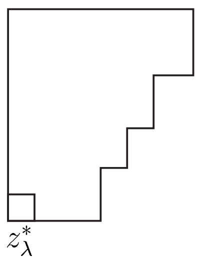


Figure 2.5: Location of cell z_λ^* in partition λ

are unique in λ and μ respectively. In particular

$$\begin{aligned} c(z_\lambda^*) &= 1-k < c(x) \ \forall x \in \lambda \setminus z_\lambda^* \\ c(z_\mu^*) &= 1-l < c(x) \ \forall x \in \lambda \setminus z_\mu^* \end{aligned}$$

By the definition of c_γ and since $c_\lambda - c_\mu \equiv 0 \bmod n$ it must be that the contributions from these content values cancel, hence $c(z^*_\lambda) = c(z^*_\mu) \Rightarrow l = k$. Thus $\lambda = (\lambda_1, \ldots, \lambda_k), \mu = (\mu_1, \ldots, \mu_k)$, i.e., the Young diagrams have the same number of rows.

Define $\lambda^{(i)} = (\lambda_{i+1}, \ldots, \lambda_k)$, $\mu^{(i)} = (\mu_{i+1}, \ldots, \mu_k)$ with $\lambda^{(0)} = \lambda$, $\mu^{(0)} = \mu$. Define $x^*_{\lambda_i}$ to be the cell in position (i, λ_i) in λ and $x^*_{\mu_i}$ to be the cell in position (i, μ_i) in μ, i.e., the last cells in each of these rows. $c(x^*_{\lambda_i})$ and $c(x^*_{\mu_i})$ are by definition unique in their respective rows, and by similar considerations to previously we see

$$\begin{aligned} c(x^*_{\lambda_i}) &= \lambda_i - i > c(x)\ \forall x \in \lambda^{(i-1)} \setminus x^*_{\lambda_i} \\ c(x^*_{\mu_i}) &= \mu_i - i > c(x)\ \forall x \in \mu^{(i-1)} \setminus x^*_{\mu_i} \end{aligned}$$

Clearly

$$\begin{aligned} c(x^*_{\lambda_1}) &= \lambda_1 - 1 > c(x)\ \forall x \in \lambda \setminus x^*_{\lambda_1} \\ c(x^*_{\mu_1}) &= \mu_1 - 1 > c(x)\ \forall x \in \mu \setminus x^*_{\mu_1}, \end{aligned}$$

and since the contribution of these contents are unique in their partitions, in order to have $c_\lambda - c_\mu \equiv 0 \bmod n$ it must be the case that $c(x^*_{\lambda_1}) = c(x^*_{\mu_1}) \Rightarrow \lambda_1 = \mu_1$. Proceeding by induction on $\lambda^{(i)}$ and $\mu^{(i)}$ and considering $c(x^*_{\lambda_i})$ and $c(x^*_{\mu_i})$ we see that $c_\lambda - c_\mu \equiv 0 \bmod n \Rightarrow \lambda_i = \mu_i,\ 1 \le i \le k \Rightarrow \lambda = \mu$. ∎

Corollary 2.5 *The expression $R_\mathcal{K}(c_\lambda, \rho, \lambda)$ is non-zero mod 2.*

Proof
The expression $R_\mathcal{K}(c_\lambda, \rho, \lambda)$ is a product of terms of the form $(c_\lambda - c_\mu)$ with $\lambda \neq \mu$. By Theorem 2.4, all of these terms will be non-zero mod 2, hence $R_\mathcal{K}(c_\lambda, \rho, \lambda)$ is non-zero mod 2. ∎

2.4 The Homfly Skein of the Annulus

The branching rules for the Homfly skein of the annulus are due to [18], while the eigenvalues of the meridian maps are due to [19].

2.4.1 Basis Elements of the Annulus

In the Homfly skein of the annulus we have elements $Q_{\lambda,\sigma}$ which are indexed by pairs of partitions (λ, σ) and form a basis for the Homfly skein of the annulus. These elements are also eigenvectors of the meridian maps $\phi_{\mathcal{C}}$, with eigenvalue

$$s_{\lambda,\sigma} = (s - s^{-1}) \left(v^{-1} \sum_{x \in \lambda} s^{2c(x)} - v \sum_{x \in \sigma} s^{-2c(x)} \right) + \frac{v^{-1} - v}{s - s^{-1}}$$

for $\phi_{\mathcal{C}}$, and eigenvalue $s_{\sigma,\lambda}$ for the meridian map $\overline{\phi_{\mathcal{C}}}$ (Figure 2.6). As with the eigenvalues of the meridian map $\phi_{\mathcal{K}}$, the eigenvalues $s_{\lambda,\sigma}$ are all distinct.

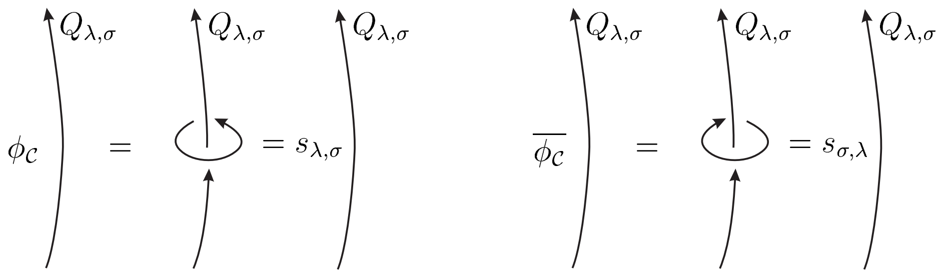


Figure 2.6: Meridian maps $\phi_{\mathcal{C}}$ and $\overline{\phi_{\mathcal{C}}}$

Note that $c_\lambda = s_{\lambda,\lambda} + 1$. We say that $Q_{\lambda,\sigma}$ is *reversible* if $\lambda = \sigma$.

2.4.2 Branching Rules

By work of Hadji we have branching rules for $Q_{\lambda,\sigma}$ [18]:

$$\begin{aligned} Q_{\rho,\epsilon} Q_{1,\emptyset} &= \sum_{\mu \in \rho^+} Q_{\mu,\epsilon} + \sum_{\tau \in \epsilon^-} Q_{\rho,\tau} \\ Q_{\rho,\epsilon} Q_{\emptyset,1} &= \sum_{\nu \in \rho^-} Q_{\nu,\epsilon} + \sum_{\delta \in \epsilon^+} Q_{\rho,\delta} \end{aligned}$$

As with the Kauffman branching rules we consider composition of two elements as diagrams in two annuli being placed one within the other. This action is commutative.

In general we want to consider $Q_{\rho,\epsilon} Q_{1,1}$ where $Q_{1,1}$ is $Q_{1,\emptyset} Q_{\emptyset,1} - Q_{\emptyset,\emptyset}$, and where $Q_{\emptyset,\emptyset}$ is the identity element, the empty diagram in the annulus. The element $Q_{1,1}$ can also be understood as the pattern for the reverse parallel satellite, as seen in Figure 1.16.

We are particularly interested in defining an analogous relation to the branching rule for $y_\rho y_1$.

By symmetry we know that

$$Q_{\rho,\rho}Q_{1,1} = \sum_{\{\mu\}} a_{\mu,\mu}Q_{\mu,\mu} + \sum_{\substack{\{(\alpha,\beta)\}\\ \alpha\neq\beta}} a_{\alpha,\beta}(Q_{\alpha,\beta}+Q_{\beta,\alpha}),$$

where $a_{\mu,\mu}$, $a_{\alpha,\beta} \in \mathbb{N}$. With the next theorem we show explicitly the branching rule for $Q_{\rho,\rho}$, and the values of $a_{\mu,\mu}$ and $a_{\alpha,\beta}$.

Theorem 2.6 *For $\lambda \in \rho^+$ we have the following relation between $Q_{\rho,\rho}$ and $Q_{\lambda,\lambda}$:*

$$Q_{\rho,\rho}Q_{1,1} = Q_{\lambda,\lambda} + \sum_{\mu\in\rho^+\cup\rho^-\setminus\lambda} Q_{\mu,\mu} + \sum_{\substack{\{(\alpha,\beta)\}\\ \alpha\neq\beta}} (Q_{\alpha,\beta}+Q_{\beta,\alpha}) + 2|\rho^-|Q_{\rho,\rho}.$$

Proof
We begin by applying the branching rules:

$$\begin{aligned}
Q_{\rho,\rho}Q_{1,1} &= Q_{\rho,\rho}(Q_{1,\emptyset}Q_{\emptyset,1} - \mathrm{id})\\
&= Q_{\rho,\rho}Q_{1,\emptyset}Q_{\emptyset,1} - Q_{\rho,\rho}\\
&= \left(\sum_{\mu\in\rho^+} Q_{\mu,\rho} + \sum_{\nu\in\rho^-} Q_{\rho,\nu}\right) Q_{\emptyset,1} - Q_{\rho,\rho}\\
&= \sum_{\mu\in\rho^+} Q_{\mu,\rho}Q_{\emptyset,1} + \sum_{\nu\in\rho^-} Q_{\rho,\nu}Q_{\emptyset,1} - Q_{\rho,\rho}\\
&= \sum_{\mu\in\rho^+}\left(\sum_{\nu\in\rho^+} Q_{\mu,\nu} + \sum_{\omega\in\mu^-} Q_{\omega,\rho}\right) + \sum_{\nu\in\rho^-}\left(\sum_{\mu\in\rho^-} Q_{\mu,\nu} + \sum_{\omega\in\nu^+} Q_{\rho,\omega}\right) - Q_{\rho,\rho}\\
&= \sum_{\mu,\nu\in\rho^+} Q_{\mu,\nu} + \sum_{\mu,\nu\in\rho^-} Q_{\mu,\nu} + \sum_{\substack{\omega\in\mu^-,\\ \mu\in\rho^+}} Q_{\omega,\rho} + \sum_{\substack{\omega\in\nu^+,\\ \nu\in\rho^-}} Q_{\rho,\omega} - Q_{\rho,\rho}
\end{aligned}$$

By definition, $\lambda \in \rho^+$ and so $Q_{\lambda,\lambda}$ is a term in the sum of $Q_{\mu,\nu}$ over ρ^+. For the sums over ρ^+ and ρ^- we extract terms where the partitions are the same:

$$\begin{aligned}\sum_{\mu,\nu\in\rho^+} Q_{\mu,\nu} + \sum_{\mu,\nu\in\rho^-} Q_{\mu,\nu} &= Q_{\lambda,\lambda} + \sum_{\mu\in\rho^+\cup\rho^-\setminus\{\lambda\}} Q_{\mu,\mu} \\ &\quad + \sum_{\substack{\alpha,\beta\in\rho^+\\ \alpha\neq\beta}} (Q_{\alpha,\beta}+Q_{\beta,\alpha}) + \sum_{\substack{\alpha,\beta\in\rho^-\\ \alpha\neq\beta}} (Q_{\alpha,\beta}+Q_{\beta,\alpha})\end{aligned}$$

A consequence of Lemma 2.2 is that $\rho^\pm \setminus \{\rho\} \equiv \rho^\mp \setminus \{\rho\}$. We use this result to split up the other sums of terms:

$$\begin{aligned}\sum_{\substack{\omega\in\mu^-,\\ \mu\in\rho^+}} Q_{\omega,\rho} &= \sum_{\omega\in\rho^\pm\setminus\{\rho\}} Q_{\omega,\rho} + |\rho^+|Q_{\rho,\rho} \\ \sum_{\substack{\omega\in\nu^+,\\ \nu\in\rho^-}} Q_{\rho,\omega} &= \sum_{\omega\in\rho^\mp\setminus\{\rho\}} Q_{\rho,\omega} + |\rho^-|Q_{\rho,\rho}\end{aligned}$$

Combining these two equations and using the result of Lemma 2.1 we can rearrange the remainder of the expression for $Q_{\rho,\rho}Q_{1,1}$:

$$\begin{aligned}\sum_{\substack{\omega\in\mu^-,\\ \mu\in\rho^+}} Q_{\omega,\rho} + \sum_{\substack{\omega\in\nu^+,\\ \nu\in\rho^-}} Q_{\rho,\omega} - Q_{\rho,\rho} &= \sum_{\omega\in\rho^\pm\setminus\{\rho\}} Q_{\omega,\rho} + |\rho^+|Q_{\rho,\rho} + \sum_{\omega\in\rho^\mp\setminus\{\rho\}} Q_{\rho,\omega} + |\rho^-|Q_{\rho,\rho} - Q_{\rho,\rho} \\ &= \sum_{\omega\in\rho^\pm\setminus\{\rho\}} (Q_{\omega,\rho}+Q_{\rho,\omega}) + (|\rho^+|+|\rho^-|-1)Q_{\rho,\rho} \\ &= \sum_{\omega\in\rho^\pm\setminus\{\rho\}} (Q_{\omega,\rho}+Q_{\rho,\omega}) + 2|\rho^-|Q_{\rho,\rho}.\end{aligned}$$

Thus we express $Q_{\rho,\rho}Q_{1,1}$ in the format desired,

$$\begin{aligned}Q_{\rho,\rho}Q_{1,1} &= Q_{\lambda,\lambda} + \sum_{\mu\in\rho^+\cup\rho^-\setminus\{\lambda\}} Q_{\mu,\mu} + \sum_{\substack{\alpha,\beta\in\rho^+\\ \alpha\neq\beta}} (Q_{\alpha,\beta}+Q_{\beta,\alpha}) \\ &\quad + \sum_{\substack{\alpha,\beta\in\rho^-\\ \alpha\neq\beta}} (Q_{\alpha,\beta}+Q_{\beta,\alpha}) + \sum_{\omega\in\rho^\pm\setminus\{\rho\}} (Q_{\omega,\rho}+Q_{\rho,\omega}) + 2|\rho^-|Q_{\rho,\rho}.\end{aligned}$$

∎

2.4.3 Note

In the proof of Theorem 2.6 we obtained the explicit details of the sets that the sums of pairs of elements are taken over. The details of these are not of great consequence in the proof of the main result of this chapter: the importance of Theorem 2.6 is showing the general relation between $Q_{\rho,\rho}$ and $Q_{\lambda,\lambda}$ for $\lambda \in \rho^+$, and showing that other terms are in the form of pairs $Q_{\alpha,\beta} + Q_{\beta,\alpha}$.

2.5 Results

In this section we introduce the theorem that motivates this chapter; this gives a correspondence between the Homfly and Kauffman polynomials of certain related links. We state the conjecture made in [35], the proof of which is the work of the remainder of this chapter. We give several other results that will be essential in this proof.

2.5.1 Rings of polynomials

It is clear that $\mathbb{Z}[v, s]$ and $\mathbb{Z}_2[v, s]$ are both rings, and that a map f between the two of them that takes integer coefficients modulo 2 is a ring homomorphism. Our situation is different because we have to account for the possibility of negative powers of v and s and also for products of denominators of the form $s^r - s^{-r}$; we must verify that the inclusion of these elements still gives a ring.

Fortunately there is a result given in [21] that guarantees this. We need to give two definitions before we can state the theorem.

Definition

For a ring R, M is a **multiplicatively closed subset** not containing 0 if $M \subseteq R$, $1 \in M$, $0 \notin M$ and M is closed under multiplication.

Let $Z(M)$ be the set $\{r \in R : rm = 0 \text{ for some } m \in M\}$.

We are now in a position to give the theorem that will allow us to confirm that the objects we wish to work with are rings.

Theorem 2.7 ([21] p.247) *Let $M \subseteq R$ be a multiplicatively closed subset, and assume that $Z(M) = 0$. Then there exists a unique overring $S \supseteq R$ such that every element of M is a unit in S and every element of S has the form rm^{-1} for some $r \in R$, $m \in M$.*

We use this theorem in the proof of the following proposition to confirm that $\mathbb{Z}[v^{\pm 1}, s^{\pm 1}]$ is a ring.

Proposition 2.8 *The set $\{\frac{a}{s^r - s^{-r}} : a \in \mathbb{Z}[v^{\pm 1}, s^{\pm 1}], r \in \mathbb{Z} \setminus \{0\}\}$ is a ring.*

Proof
We know $R = \mathbb{Z}[v, s]$ is a ring. Take a subset M of R defined as

$$M = \{v^m s^n \prod_{i=0}^{k} (s^{r_i} - s^{-r_i}) : m, n \in \mathbb{Z},\ r_i \in \mathbb{Z} \setminus \{0\},\ k \in \mathbb{N}\}.$$

Clearly M is closed under multiplication, and $1 \in M$, $0 \notin M$, hence by Theorem 2.7 there is an overring S such that every element has the form rm^{-1} for some $r \in \mathbb{Z}[v, s]$, $m \in M$. Then

$$\begin{aligned} S &= \{rm^{-1} : r \in \mathbb{Z}[v, s], m \in M\} \\ &= \{\frac{a}{s^r - s^{-r}} : a \in \mathbb{Z}[v^{\pm 1}, s^{\pm 1}], r \in \mathbb{Z} \setminus \{0\}\}. \end{aligned}$$

∎

S is the object we have previously denoted as $\mathbb{Z}[v^{\pm 1}, s^{\pm 1}]$. Similarly, we can show that the object we have denoted $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ is a ring.

A map $f : \mathbb{Z}[v^{\pm 1}, s^{\pm 1}] \to \mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ where the integer coefficients of the polynomial are reduced modulo 2 is clearly a homomorphism between the rings.

Having cleared up the status of the rings that we will work in, we now consider some other important results that we will need in the main proof of this chapter. We begin with the result of Rudolph which motivates the more general result that we wish to show.

2.5.2 Satellites and patterns

Theorem 2.9 (Rudolph [54]) *The Kauffman polynomial of a link with substitution $v, s \to v^2, s^2$ and taking coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ is the same as the Homfly polynomial of its reverse parallel satellite taking coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$, with the empty diagram taking the normalisation of 1 for both invariants.*

Note that for this result and for others in this chapter we take a different normalisation to those given in Chapter 1.

Morton's conjecture [35] offers a much greater generalisation of this theorem, by allowing us a much greater degree of freedom in decorating the components of the link. Recall that in Section 1.7 we defined decorating a knot with a pattern from the annulus. In the case of Conjecture 2.10 we (potentially) decorate each link component with a different pattern.

Conjecture 2.10 (Morton [35]) *Decorate each component L_i of a framed unoriented link L by $y_{\lambda(i)}$. The Kauffman polynomial of this decorated link with substitution $v, s \to v^2, s^2$ and taking coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ is the same as the Homfly polynomial of L when each L_i is decorated by $Q_{\lambda(i),\lambda(i)}$ taking coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$, with the empty diagram taking the normalisation of 1 for both invariants.*

In light of this conjecture we restate Theorem 2.9 as follows:
Restatement of Theorem 2.9 *Decorate each component L_i of a framed unoriented link L by y_1. The Kauffman polynomial of this decorated link with substitution $v, s \to v^2, s^2$ and taking coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ is the same as the Homfly polynomial of L when each L_i is decorated by $Q_{1,1}$ taking coefficients $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$, with the empty diagram taking the normalisation of 1 for both invariants.*

Before considering the branching rules in the Homfly skein of the annulus again, there is one more result that we need for our proof of Conjecture 2.10. For satellites of decorated by certain linear combinations of patterns from the

Homfly skein of the annulus, we show that we are able to dispose of certain parts of the pattern without affecting the invariant modulo 2.

Lemma 2.11 *Decorate each component L_i of a link L with linear combinations of patterns from the Homfly skein of the annulus of the form*

$$P_i = \sum_{\{(\alpha_i,\beta_i)\}} (Q_{\alpha_i,\beta_i} + Q_{\beta_i,\alpha_i}) + \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i}.$$

With coefficients from $\mathbb{Z}_2[v^{\pm}, s^{\pm}]$, the Homfly polynomial of L where each component L_i decorated by P_i is the same as the Homfly polynomial of L where each component L_i decorated by

$$P_i' = \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i}.$$

Proof
Consider a pair of partitions $({\alpha_i}^*, {\beta_i}^*) \in \{(\alpha_i, \beta_i)\}$ and write P_i as

$$P_i = Q_{{\alpha_i}^*,{\beta_i}^*} + Q_{{\beta_i}^*,{\alpha_i}^*} + \sum_{\substack{\{(\alpha_i,\beta_i)\} \\ (\alpha_i,\beta_i)\neq({\alpha_i}^*,{\beta_i}^*)}} (Q_{\alpha_i,\beta_i} + Q_{\beta_i,\alpha_i}) + \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i}.$$

Fix patterns P_j on all other components L_j of the link L. The Homfly polynomial of L with decorations P_j on components L_j and the decoration P_i on L_i is equal to the sum of the Homfly polynomials of L with decorations P_j on L_j and each term in P_i counted separately on L_i. Consider the Homfly polynomial of L with P_j on L_j and $Q_{{\alpha_i}^*,{\beta_i}^*}$ on L_i. By Lemma 1.6, reversing orientations of all components leaves the Homfly polynomial unchanged and leaves the patterns P_j unchanged, but the pattern on L_i becomes $Q_{{\beta_i}^*,{\alpha_i}^*}$.

Hence the Homfly polynomial of L with patterns P_j on components L_j and $Q_{{\alpha_i}^*,{\beta_i}^*}$ on L_i is equal to the Homfly polynomial of L with patterns P_j on components L_j and $Q_{\beta_i{}^*,\alpha_i{}^*}$ on L_i. Thus

$$\begin{aligned} P_i &= \sum_{\{(\alpha_i,\beta_i)\}} (Q_{\alpha_i,\beta_i} + Q_{\beta_i,\alpha_i}) + \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i} \\ &= \sum_{\{(\alpha_i,\beta_i)\}} Q_{\alpha_i,\beta_i} + \sum_{\{(\alpha_i,\beta_i)\}} Q_{\beta_i,\alpha_i} + \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i} \\ &= \sum_{\{(\alpha_i,\beta_i)\}} Q_{\alpha_i,\beta_i} + \sum_{\{(\alpha_i,\beta_i)\}} Q_{\alpha_i,\beta_i} + \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i} \\ &= 2\sum_{\{(\alpha_i,\beta_i)\}} Q_{\alpha_i,\beta_i} + \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i} \\ &\equiv \sum_{\{\theta_i\}} Q_{\theta_i,\theta_i} \bmod 2 \\ &\equiv P_i' \bmod 2 \end{aligned}$$

where mod 2 denotes taking coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$. ∎

2.6 More in the Homfly Skein of the Annulus

We return to considering the multiplication $Q_{\rho,\rho}Q_{1,1}$. By Theorem 2.6 we have evaluated this as

$$Q_{\rho,\rho}Q_{1,1} = Q_{\lambda,\lambda} + \sum_{\mu\in\rho^+\cup\rho^-\setminus\{\lambda\}} Q_{\mu,\mu} + \sum_{\{(\alpha,\beta)\}} (Q_{\alpha,\beta} + Q_{\beta,\alpha}) + 2|\rho^-|Q_{\rho,\rho}.$$

As with the expression for y_λ we wish to eliminate the sum of terms in $Q_{\mu,\mu}$ from the expression.

Definition

For partitions ρ and λ, $\lambda \in \rho^+$, define the polynomial $R_{\mathcal{C}}(t,\rho,\lambda)$ by

$$R_{\mathcal{C}}(t,\rho,\lambda) = \prod_{\mu\in\rho^+\cup\rho^-\setminus\{\lambda\}} (t - ({s_{\mu,\mu}}^2 - 1)).$$

Let $\phi_{\mathcal{C}}\overline{\phi_{\mathcal{C}}} - 1$ be the map $\Phi_{1,1}$; elements $Q_{\gamma,\theta}$ and $Q_{\theta,\gamma}$ have eigenvalue $s_{\gamma,\theta}s_{\theta,\gamma} - 1$ for $\Phi_{1,1}$.

Lemma 2.12 *For partitions α, β, $\alpha \neq \beta$*

$$R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)(Q_{\alpha,\beta} + Q_{\beta,\alpha}) = R_{\mathcal{C}}(s_{\alpha,\beta}s_{\beta,\alpha} - 1,\rho,\lambda)(Q_{\alpha,\beta} + Q_{\beta,\alpha}).$$

Proof

$$\begin{aligned}
R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)(Q_{\alpha,\beta}+Q_{\beta,\alpha}) &= R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)(Q_{\alpha,\beta})+R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)(Q_{\beta,\alpha}) \\
&= R_{\mathcal{C}}(s_{\alpha,\beta}s_{\beta,\alpha}-1,\rho,\lambda)Q_{\alpha,\beta} \\
&\quad +R_{\mathcal{C}}(s_{\beta,\alpha}s_{\alpha,\beta}-1,\rho,\lambda)Q_{\beta,\alpha} \\
&= R_{\mathcal{C}}(s_{\alpha,\beta}s_{\beta,\alpha}-1,\rho,\lambda)(Q_{\alpha,\beta}+Q_{\beta,\alpha}).
\end{aligned}$$

∎

Lemma 2.12 shows that the coefficient of each element in a sum $Q_{\alpha,\beta}+Q_{\beta,\alpha}$ remains equal after we apply $R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)$.

The next step in our construction is to apply $R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)$ to the relation we have already derived from the branching rules.

Lemma 2.13

$$\begin{aligned}
R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)(Q_{\rho,\rho}Q_{1,1}) &= R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)Q_{\lambda,\lambda}+2|\rho^-|R_{\mathcal{C}}(s^2_{\rho,\rho}-1,\rho,\lambda)Q_{\rho,\rho} \\
&\quad +\sum_{\{(\alpha,\beta)\}} R_{\mathcal{C}}(s_{\alpha,\beta}s_{\beta,\alpha}-1,\rho,\lambda)(Q_{\alpha,\beta}+Q_{\beta,\alpha}).
\end{aligned}$$

Proof
This follows almost immediately from applying $R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)$ to the relation from Theorem 2.6. The sum of elements $Q_{\mu,\mu}$ over $\rho^+\cup\rho^-\setminus\{\lambda\}$ is cancelled out by applying $R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)$. Using Lemma 2.12 on the sum of terms $Q_{\alpha,\beta}+Q_{\beta,\alpha}$ and the remaining terms gives the coefficients indicated. ∎

By the symmetry we observed earlier we note that

$$\left(\sum_{\{(\tau,\gamma)\}}(Q_{\tau,\gamma}+Q_{\gamma,\tau})\right)Q_{1,1}=\sum_{\{(\alpha,\beta)\}}(Q_{\alpha,\beta}+Q_{\beta,\alpha}),$$

i.e., multiplying a sum of terms $Q_{\tau,\gamma}+Q_{\gamma,\tau}$ by $Q_{1,1}$ results in a similar sum of terms.

2.7 Proving Conjecture 2.10

We prove the conjecture by induction, and use the constructions that we have already noted for y_λ, $Q_{\lambda,\lambda}$, $R_{\mathcal{K}}(t,\rho,\lambda)$ and $R_{\mathcal{C}}(t,\rho,\lambda)$ to draw out correspondences between the two sets of branching rules when we consider coefficients from $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$.

Proof

Take a link L with l components $L_1,\ldots,L_l$. Let $L(\lambda(1),\ldots,\lambda(l))$ denote the link L with each component L_i paired with a partition $\lambda(i)$.

Let

$$N=\sum_{i=1}^{l}(|\lambda(i)|-1) \quad , \quad |\lambda(i)|\geq 1,$$

and we use this to base our induction on. By definition $N\geq 0$ and the only case when $N=0$ is when $|\lambda(i)|=1$ for all i. This is the situation when the patterns decorating each component are y_1 in the Kauffman skein of the annulus and $Q_{1,1}$ in the Homfly skein of the annulus. By Theorem 2.9 we know that the case $N=0$ satisfies the conditions of the conjecture, and thus provides a basis for our proof by induction.

Assume that for all $N\leq n-1$ the conjecture is true. For any case when $N=n$ we know that only one partition $\lambda(i)$ is different from some case when $N=n-1$, and it is different by the addition of only one cell to that partition, $\rho(i)$. Without loss of generality, we can assume that the partition that has changed is attached to component L_l. Effectively the difference between the two links resulting from the attached partitions is $\rho(l)$ paired with L_l when $N=n-1$ and $\lambda(l)$ paired with L_l when $N=n$.

As the difference between $\lambda(l):=\lambda$ and $\rho(l):=\rho$ is one cell then we know $\rho\subset\lambda$, $|\lambda|=|\rho|+1$. Thus, when we decorate the link either in the Kauffman or Homfly skein we can use the branching rules to find expressions for y_λ and $Q_{\lambda,\lambda}$ in terms of y_ρ and $Q_{\rho,\rho}$ respectively.

By Lemma 2.3 we know that y_λ can be expressed as a certain linear combination of longitudes and meridians, but we now need to show that this is in alignment under the conditions of the conjecture with the more complicated expression that we have for the Homfly case.

Due to the way that we are building up patterns we must (at least at this stage) include the possibility that there are pairs of patterns (as we have defined them previously) which are also going to be multiplied by $Q_{1,1}$. However, by the note that we made before, this will itself only contribute another sum over pairs of patterns in the branching rule. By Theorem 2.6 and Lemma 2.13 the expression that we need to consider is

$$\begin{aligned} R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)\left(\left(Q_{\rho,\rho}+\sum_{\{(\tau,\gamma)\}}(Q_{\tau,\gamma}+Q_{\gamma,\tau})\right)Q_{1,1}\right) \\ = R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)Q_{\lambda,\lambda}+2|\rho^-|R_{\mathcal{C}}(s^2_{\rho,\rho}-1)Q_{\rho,\rho} \\ +\sum_{\{(\alpha,\beta)\}}R_{\mathcal{C}}(s_{\alpha,\beta}s_{\beta,\alpha}-1,\rho,\lambda)(Q_{\alpha,\beta}+Q_{\beta,\alpha}). \end{aligned}$$

There are differences in the skeins between the branching rules for y_λ and $Q_{\lambda,\lambda}$. If we work modulo 2 and with the substitution $v,s\to v^2,s^2$ for Kauffman there is no immediate change that we can observe.

However, working modulo 2 for Homfly we simplify the expression that we have for $Q_{\lambda,\lambda}$: the term of $Q_{\rho,\rho}$ on the right hand side obviously vanishes mod 2. The sums of pairs of patterns cancel by Lemma 2.11 since all of the other components in the link L are being decorated by patterns $Q_{\rho(j),\rho(j)}$ for fixed partitions $\rho(j)$. Hence mod 2 we have

$$R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)\left(\left(Q_{\rho,\rho}+\sum_{\{(\tau,\gamma)\}}(Q_{\tau,\gamma}+Q_{\gamma,\tau})\right)Q_{1,1}\right)=R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)Q_{\lambda,\lambda}$$

in the Homfly skein of the annulus. We can go a step further and eliminate the sum of pairs on the left hand side of the expression to give

$$R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)(Q_{\rho,\rho}Q_{1,1})=R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)Q_{\lambda,\lambda}.$$

From similar considerations to Theorem 2.4 and Corollary 2.5 it can be shown that $R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)$ is non-zero mod 2, and hence we can give the following construction for elements in the Homfly skein of the annulus when we consider coefficients mod 2:

$$Q_{\lambda,\lambda} = \frac{R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)}{R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)}(Q_{\rho,\rho}Q_{1,1}).$$

There is clearly a parallel between the expressions for the two branching rules. By the assumption of the inductive argument the decoration of $Q_{\rho,\rho}$ and y_ρ agree, and by the result of Rudolph we know that $Q_{1,1}$ and y_1 agree.

To prove the conjecture we must show that $R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)$ and $R_{\mathcal{K}}(\phi_{\mathcal{K}},\rho,\lambda)$ agree, and that $R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)$ and $R_{\mathcal{K}}(c_\lambda,\rho,\lambda)$ agree under the conditions of the conjecture. Recall

$$\begin{aligned} R_{\mathcal{K}}(t,\rho,\lambda) &= \prod_{\mu\in\rho^+\cup\rho^-\setminus\{\lambda\}} (t-c_\mu) \\ R_{\mathcal{C}}(t,\rho,\lambda) &= \prod_{\mu\in\rho^+\cup\rho^-\setminus\{\lambda\}} (t-({s_{\mu,\mu}}^2-1)) \end{aligned}$$

which are both defined as products over the same set of partitions, with the difference being the factors in each. A typical factor in $R_{\mathcal{K}}(t,\rho,\lambda)$ is $(t-c_\gamma)$ and a typical factor in $R_{\mathcal{C}}(t,\rho,\lambda)$ is $(t-({s_{\gamma,\gamma}}^2-1))$. In the first instance we need to compare c_γ and ${s_{\gamma,\gamma}}^2-1$ under the conditions of the conjecture.

For the Kauffman polynomial $D(v,s)$ the substitution $v,s\to v^2,s^2$ modulo 2 is equivalent to squaring the polynomial modulo 2, i.e.,

$$D(v^2,s^2)\equiv(D(v,s))^2 \bmod 2.$$

We noted previously that $c_\gamma = s_{\gamma,\gamma}+1$. Under the substitution c_γ is equivalent to ${c_\gamma}^2$ mod 2. Then

$$\begin{aligned} {c_\gamma}^2 &\equiv (s_{\gamma,\gamma}+1)^2 \bmod 2 \\ &\equiv {s_{\gamma,\gamma}}^2+2s_{\gamma,\gamma}+1 \bmod 2 \\ &\equiv {s_{\gamma,\gamma}}^2-1 \bmod 2, \end{aligned}$$

as required. Thus $R_{\mathcal{C}}(s^2_{\lambda,\lambda}-1,\rho,\lambda)\equiv R_{\mathcal{K}}(c_\lambda,\rho,\lambda)$ under the conditions of the conjecture, and for $R_{\mathcal{C}}(\Phi_{1,1},\rho,\lambda)$ and $R_{\mathcal{K}}(\phi_{\mathcal{K}},\rho,\lambda)$ we need only note that

$$\Phi_{1,1} = \phi_{\mathcal{C}}\overline{\phi_{\mathcal{C}}}-1 = \phi_{\mathcal{K}} * Q_{1,1}$$

and so the meridian maps and coefficients from eigenvalues of the meridian maps agree, as required. ∎

Owing to the fact that $D(v^2, s^2) \equiv (D(v,s))^2$ mod 2 we can then state the theorem as follows:

Theorem 2.14 *Decorate each component L_i of a framed unoriented link L by $y_{\lambda(i)}$. The square of the Kauffman polynomial of this decorated link with coefficients in $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$ is equal to the Homfly polynomial of L when each L_i is decorated by $Q_{\lambda(i),\lambda(i)}$ with coefficients in $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$, with the empty diagram taking the normalisation of 1 for both invariants.*

There is a fairly neat corollary that we can give to Theorem 2.14, for the situation that we want to take linear combinations of patterns when we decorate our links. Before stating and proving it, it is in our best interests to introduce some notation so that we can give the corollary and proof as simply as possible.

Definition

Let $L(\Lambda(1), \ldots, \Lambda(l))$ denote the link L with each component L_i paired with a set of partitions $\Lambda(i) = \{\lambda(i_1), \ldots, \lambda(i_j)\}$, where $j = |\Lambda(i)|$.

Then let $L^{\mathcal{K}}(\Lambda(1), \ldots, \Lambda(l))$ denote the link L with each component L_i decorated by a linear combination of patterns

$$Y_{\Lambda(i)} = y_{\lambda(i_1)} + \ldots + y_{\lambda(i_j)}$$

and let $L^{\mathcal{C}}(\Lambda(1), \ldots, \Lambda(l))$ denote the link L with each component L_i decorated by a linear combination of patterns

$$S_{\Lambda(i)} = Q_{\lambda(i_1),\lambda(i_1)} + \ldots + Q_{\lambda(i_j),\lambda(i_j)}.$$

Finally, take $D(L^{\mathcal{K}}(\Lambda(1), \ldots, \Lambda(l)))$ to denote the Kauffman polynomial of $L^{\mathcal{K}}(\Lambda(1), \ldots, \Lambda(l))$, and let $P(L^{\mathcal{C}}(\Lambda(1), \ldots, \Lambda(l)))$ denote the Homfly polynomial of $L^{\mathcal{C}}(\Lambda(1), \ldots, \Lambda(l))$.

Corollary 2.15 $D(L^{\mathcal{K}}(\Lambda(1), \ldots, \Lambda(l)))^2 \equiv P(L^{\mathcal{C}}(\Lambda(1), \ldots, \Lambda(l)))$ *mod 2, i.e., taking coefficients $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$, and with the empty diagram taking the normalisation of 1 for both invariants.*

Proof
The Kauffman polynomial of a satellite decorated by a linear combination of patterns is equal to the sum of the Kauffman polynomials of the link if it is decorated by each pattern separately; a similar statement can be made about the Homfly polynomial of satellites decorated in such a way.

In the case that we are considering, as a first step we can state the following:

$$\begin{aligned} D(L^{\mathcal{K}}(\Lambda(1),\ldots,\Lambda(l))) &= D(L^{\mathcal{K}}(\lambda(1_1),\ldots,\Lambda(l))) + \ldots + D(L^{\mathcal{K}}(\lambda(1_j),\ldots,\Lambda(l))) \\ &= \sum_{k=1}^{|\Lambda(1)|} D(L^{\mathcal{K}}(\lambda(1_k),\ldots,\Lambda(l))) \\ &= \sum_{m=1}^{l}\sum_{k=1}^{|\Lambda(m)|} D(L^{\mathcal{K}}(\ldots,\lambda(m_k),\ldots)). \end{aligned}$$

Then with coefficients in $\mathbb{Z}_2[v^{\pm 1}, s^{\pm 1}]$,

$$\begin{aligned} D(L^{\mathcal{K}}(\Lambda(1),\ldots,\Lambda(l)))^2 &\equiv D(L^{\mathcal{K}}(\Lambda(1),\ldots,\Lambda(l)))_{|{v\to v^2 \atop s\to s^2}} \\ &\equiv \sum_{m=1}^{l}\sum_{k=1}^{|\Lambda(m)|} D(L^{\mathcal{K}}(\ldots,\lambda(m_k),\ldots))_{|{v\to v^2 \atop s\to s^2}} \\ &\equiv \sum_{m=1}^{l}\sum_{k=1}^{|\Lambda(m)|} P(L^{\mathcal{C}}(\ldots,\lambda(m_k),\ldots)) \\ &\equiv P(L^{\mathcal{C}}(\Lambda(1),\ldots,\Lambda(l))). \end{aligned}$$

∎

Chapter 3

Stacked k-tangles and the Kauffman Polynomial

3.1 Introduction

In this chapter we give details of an algorithm for calculating the Kauffman polynomial of a link. As noted previously, calculating knot polynomials from skein relations generally gives an exponential time algorithm based on the number of crossings in the diagram that we start with.

Przytycki showed that a polynomial time algorithm was possible in principle for the Kauffman polynomial [49], although the method he gave only calculates a part of the coefficients for the Kauffman polynomials. The work of this chapter presents the first complete polynomial time algorithm for calculating the Kauffman polynomial.

We explore k-tangles and stacked k-tangles, and how we compose a stacked k-tangle with a word from the braid group $\mathbb{B}_{2k}$. By representing stacked k-tangles as k-sequences, and then exploring conditions that guarantee a desired outcome we obtain the foundation of the algorithm that we construct (leading to Proposition 3.8).

3.2 Stacked k-tangles

Tangle diagrams are often given in terms of inputs and outputs to a box, with the arcs inside being knotted somehow.

Definition

An (m, n)**-tangle** is a box with m inputs at the top of the box and n outputs at the bottom, where $m + n = 2l$ for some l. Connecting the $m + n$ points are l arcs, and these can be freely knotted inside the box. We also allow closed components in the tangle box.

Definition

A k**-tangle** consists of k arcs connected to $2k$ points on a line, with arcs lying in the upper half space and each having a single local maximum. There are no restrictions on how the arcs lie relative to each other, but we do not allow closed components within the tangle.

Essentially, a k-tangle is a $(0, 2k)$-tangle with unknotted arcs and the extra condition not allowing closed components. A consequence of the definition of a k-tangle is that the arcs are all individually knotted. See Figure 3.1 for examples of 3- and 4-tangles.

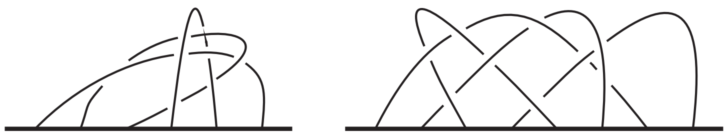


Figure 3.1: Examples of 3- and 4-tangles

A k-tangle can be drawn as a $2k$ braid with a plait closure of k caps at the top.

We now give the most important definition of the next two chapters; the methods that we will begin to outline shortly depend on this definition and its consequences.

Definition

A **stacked k-tangle** is a k-tangle such that arcs do not wind around each other, i.e., no two arcs are linked.

Figure 3.2 shows some examples of stacked 4-tangles, and as before note that we do not allow the possibility of closed components in the stacked tangles.

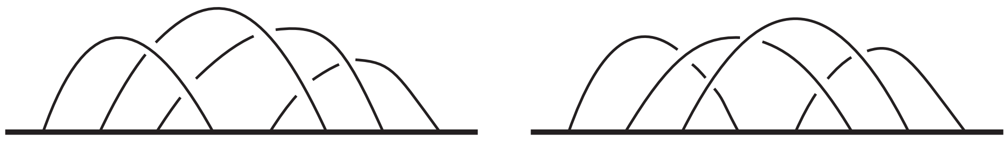


Figure 3.2: Examples of stacked 4-tangles

A variation on this definition was originally given in [41].

As the arcs are stacked, we can consider them as being in separate layers, and then give a numbering to these arcs. The top-most arc is numbered 1, and the bottom-most is numbered k, with the arcs inbetween numbered according to the rule that an overcrossing arc has a lower number than the arc it crosses over. For example, we number the stacked 4-tangle to the left in Figure 3.2 as in Figure 3.3.

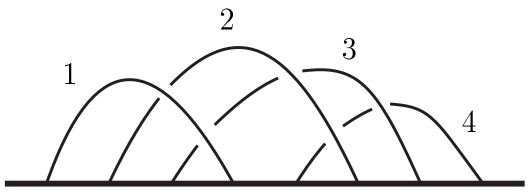


Figure 3.3: Numbering arcs of a stacked 4-tangle

There is not necessarily a unique numbering for the arcs of a diagram; it is clear that we can have stacked k-tangles which have two or more arcs in the same layer. In this case we need only give the arcs a numbering so that they respect whatever arcs might lie above or below them in the diagram.

Consider the stacked 4-tangle to the right of Figure 3.2: this has two possible numberings, as can be seen in Figure 3.4.

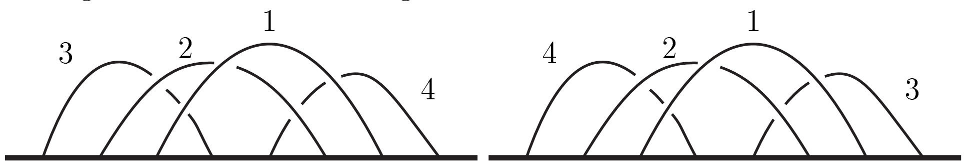


Figure 3.4: A stacked 4-tangle without a unique numbering

By considering the numbers of the endpoints of the arcs of a stacked tangle we see that this information determines the diagram. Reading these numbers from left to right we use the number sequence to represent the stacked tangle.

Definition

A **k-sequence** is a sequence of numbers representing the endpoints of the arcs of a stacked k-tangle; the k-sequence determines the stacked k-tangle.

For example, the 4-sequence for the stacked 4-tangle in Figure 3.3 is (12314234), and the two possible number sequences for the stacked 4-tangle in Figure 3.4 are (32134214) and (42143213).

As there are stacked k-tangles without unique k-sequences determining them, it is clear that the number of k-sequences will be greater than the number of stacked k-tangles for $k \geq 2$.

Proposition 3.1 *The set of k-sequences has $\frac{(2k)!}{2^k}$ elements.*

Proof

The number of elements in the set of k-sequences is easily calculable from simple combinatorics. We permute $2k$ objects – but there are k distinct objects, each of which occurs twice. Hence the number of elements is $\frac{(2k)!}{2^k}$. ∎

Calculating the size of the set of stacked k-tangles is more complicated, and it is less clear if there is a simple way to do this in general. We will consider this problem further in Section 3.8.

3.3 Multiplying stacked tangles by braids

Our aim in this section is to express a general k-tangle as a linear combination of stacked k-tangles with respect to the Kauffman skein relations. We work in the Kauffman skein module of stacked k-tangles.

As stated previously, a k-tangle can be expressed as a $2k$ braid with a plait closure at the top. We can also consider a k-tangle as a stacked k-tangle composed from below with a word from $\mathbb{B}_{2k}$. In both cases we do this in an obvious way, by pulling the arcs, lengthening the diagram until we can see a stacked k-tangle composed with a braid.

We consider this idea in the example of Figure 3.5, which is taken from the 3-tangle of Figure 3.1.

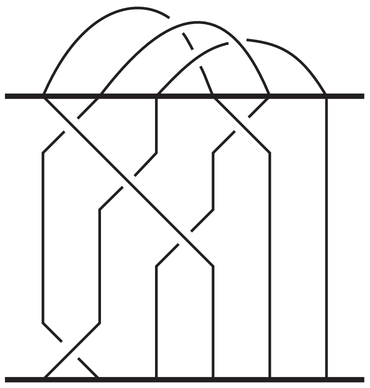


Figure 3.5: Multiplying a stacked 3-tangle by a braid

By considering diagrams of this type we begin to examine what we mean by multiplying a stacked tangle by a braid word in the skein. We consider this as the action of the braid group $\mathbb{B}_{2k}$ on the Kauffman skein module of stacked k-tangles.

We start by considering what happens when we multiply a stacked tangle by a braid generator. In order to give a consistent system for this, we give conditions for when multiplication by a braid generator results in a stacked tangle.

Definition

A stacked k-tangle t_1 is **compatible** with a braid generator σ_i if the action of multiplying t_1 by σ_i results in another stacked k-tangle t_2 or t_1 multiplied by a scalar v. Similarly, a stacked k-tangle t_1 is compatible with an inverse σ_i^{-1} if the action of multiplying by the inverse results in another stacked k-tangle t_2 or t_1 multiplied by a scalar v^{-1}.

We use k-sequences to represent stacked k-tangles and so must give a statement as to how we consider compatibility with respect to k-sequences.

Definition

A k-sequence s is compatible with a generator σ_i if the stacked k-tangle defined by s is compatible with σ_i. Similarly, s is compatible with an inverse σ_i^{-1} if the stacked k-tangle defined by s is compatible with σ_i^{-1}.

The following proposition gives conditions that ensure a k-sequences is compatible with a given generator.

Proposition 3.2 *If the number at position i in a k-sequence s is greater than or equal to the number at position $i+1$ then s is compatible with σ_i.*

Proof

Let $s(j)$ stand for the number at position j in the k-sequence. If $s(i) = s(i+1)$ then s is compatible with σ_i as the two positions in s represent the two ends of one arc. This arc results in the original stacked k-tangle being multiplied by v in order to remove a kink by a Type I Reidemeister move.

If $s(i) > s(i+1)$ then this means that the arc which has an endpoint at $s(i+1)$ is numbered in such a way that it is considered to be above the arc which has an endpoint at $s(i)$. Thus, s is compatible with σ_i as the action of the generator brings the lower-numbered arc across the higher-numbered. ∎

We can state something similar when dealing with inverses.

Proposition 3.3 *If the number at position i in a k-sequence s is less than or equal to the number at position $i+1$ then s is compatible with σ_i^{-1}.*

For the purpose of constructing the algorithm, in all following references to compatibility we assume that we describe k-sequences that satisfy the conditions of Propositions 3.2 and 3.3.

3.3.1 Renumbering

We have previously noted that a stacked k-tangle can have more than one valid k-sequence that defines it.

Definition

Let s and s' be k-sequences. We say that s is *equivalent by renumbering* to s' if and only if s' arises from a valid numbering for the same stacked k-tangle that s determines.

We will consider renumbering as an essential relation for the purpose of obtaining k-sequences which satisfy the compatibility conditions of Propositions 3.2 and 3.3. The following results give the foundation for showing when renumbering is possible.

Proposition 3.4 *Let $1 \leq a \leq k-1$ and $b = a+1$. Consider two k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = a & s_1(r) = b & s_1(t) = b \\ s_2(p) = b & s_2(q) = b & s_2(r) = a & s_2(t) = a \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

Proof

In the stacked k-tangle determined by s_1 we would consider the arc numbered a as being immediately "above" arc b. Regardless of how the other arcs lie relative to a and b, we consider a and b as in the left-hand diagram in Figure 3.6, i.e., they do not cross. The numbering of s_2 would result in the same situation, and hence s_1 and s_2 are related by renumbering. ∎

Proposition 3.5 *Let $1 \leq a \leq k-1$ and $b = a+1$. Consider two k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = b & s_1(r) = b & s_1(t) = a \\ s_2(p) = b & s_2(q) = a & s_2(r) = a & s_2(t) = b \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

Proof
As with the previous proof, regardless of how the other arcs in the stacked k-tangle defined by s_1 lie relative to a and b, we consider a and b as in the right-hand diagram in Figure 3.6, i.e., they do not cross. The numbering of s_2 would result in the same situation, and hence s_1 and s_2 are related by renumbering. ∎

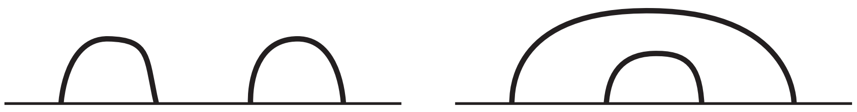


Figure 3.6: Non-crossing arcs

We say that a and b are *adjacent* where $1 \leq a \leq k-1$ and $b = a+1$.

3.3.2 Rearrangement

Consider the diagram of Figure 3.7. This shows the stacked 3-tangle given by

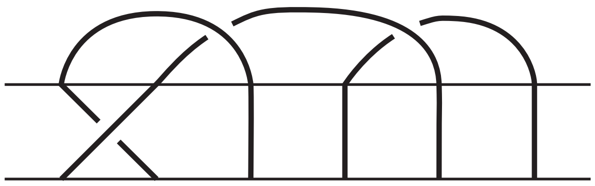


Figure 3.7: (121323) multiplied by σ_1

3-sequence (121323) being multiplied by a generator σ_1 that is incompatible.

Definition

A k-sequence s is *Kauffman equivalent* to $\sum a_i s_i$, a linear combination of k-sequences with coefficients from the Kauffman skein module, if and only if a linear combination of stacked k-tangles corresponding to the linear combination of k-sequences can be obtained from the stacked k-tangle defined by s by applying the Kauffman skein relation.

We say that we use a *rearrangement* action when applying Kauffman skein relations in order to obtain a Kauffman equivalent linear combination of k-sequences.

It follows that two k-sequences equivalent by renumbering are Kauffman equivalent.

Proposition 3.6 *Let $1 \leq a \leq k-1$ and $b = a+1$. Consider four k-sequences s_1, s_2, s_3, s_4 such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = b & s_1(r) = a & s_1(t) = b \\ s_2(p) = b & s_2(q) = a & s_2(r) = b & s_2(t) = a \\ s_3(p) = a & s_3(q) = a & s_3(r) = b & s_3(t) = b \\ s_4(p) = a & s_4(q) = b & s_4(r) = b & s_4(t) = a \end{array}$$

and for all other $1 \leq i \leq 2k$ $s_1(i) = s_2(i) = s_3(i) = s_4(i)$.

Then s_1 is Kauffman equivalent to $s_2 - zs_3 + zs_4$.

Proof

We consider the occurrences of a and b within the four k-sequences as 2-sequences. We can write these as

$$s'_1 = (abab) \quad s'_2 = (baba) \quad s'_3 = (aabb) \quad s'_4 = (abba)$$

and by considering the stacked 2-tangles that they determine and the main Kauffman skein relation we can state s'_1 is Kauffman equivalent to $s'_2 - zs'_3 + zs'_4$.

By comparing this with the k-sequences that we started with, and because this rearrangement will not effect the other arcs in the stacked k-tangles that lie above or below arcs a and b we can thus state that s_1 is Kauffman equivalent to $s_2 - zs_3 + zs_4$. ∎

Note that the terms s'_3 and s'_4 can be renumbered, and hence the larger k-sequences s_3 and s_4 can be renumbered.

Proposition 3.7 *A k-sequence s with the number r at position j, where $2 \leq r \leq k$ is Kauffman equivalent to a linear combination of k-sequences each with $r-1$ at position j.*

Proof
If the numbers r and $r-1$ in s represent arcs which do not cross then Propositions 3.4 and 3.5 guarantee that there is a k-sequence s' such that $s'(j) = r-1$.

If the numbers r and $r-1$ in s represent arcs which cross then Proposition 3.6 allows us to express s as a linear combination of three k-sequences, s'_1, s'_2, s'_3, such that $s'_1(j) = s'_2(j) = s'_3(j) = r-1$. ∎

In order to be consistent let us say that we always act to reduce the number of a larger numbered endpoint in a k-sequence in an effort to make it satisfy the conditions of Proposition 3.2 or 3.3.

Proposition 3.8 *A k-sequence s that is incompatible with a generator or inverse $\sigma_i^{\pm 1}$ is Kauffman equivalent to a linear combination of k-sequences compatible with $\sigma_i^{\pm 1}$.*

Proof
This follows from Proposition 3.7 by repeated application of the result on the relevant position j in the linear combination of k-sequences. ∎

For example, consider again the diagram of Figure 3.7, which we can represent as (121323) being multiplied by σ_1. We need to rearrange the adjacent arcs 1 and 2 according to the relation we defined. Hence, by Proposition 3.6

$$(212313) - (121323) = z((112323) - (122313)).$$

The final term does not satisfy Proposition 3.7, but we renumber the 3-sequence as arcs 1 and 2 in the diagram it determines do not cross, and hence obtain the following linear combination of 3-sequences for (121323) that are compatible with σ_1,

$$(121323) = (212313) - z(112323) + z(211323)$$

and thus

$$\begin{aligned}(121323)\sigma_1 &= (212313)\sigma_1 - z(112323)\sigma_1 + z(211323)\sigma_1 \\ &= (122313) - vz(112323) + z(121323).\end{aligned}$$

Consider the diagram in Figure 3.8. This can (only) be represented as the

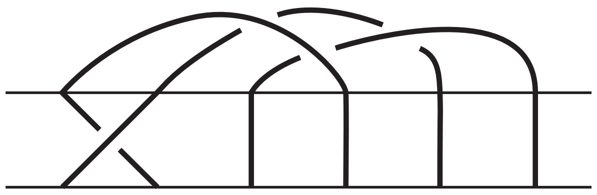


Figure 3.8: (132132) multiplied by σ_1

3-sequence (132132) multiplied by σ_1. We cannot renumber the 3-sequence so we must rearrange. In this case we need to reduce the second number in the sequence, and have to perform two sets of rearrangement; the first involves arcs 2 and 3, and the second involves possibly several rearrangement actions on arcs 1 and 2. Some of the resulting 3-sequences from the first rearrangement will only require renumbering so that they are compatible.

This example underlines the fact that we can require several acts of rearrangement and renumbering in order to express an incompatible k-sequence as a linear combination of compatible k-sequences.

Before we move on to consider the algorithm, let us formally state the relation between the k-sequences and the stacked k-tangles now that we have introduced structure from the renumbering and Kauffman equivalence.

Proposition 3.9 *The module of k-sequences modulo renumbering and Kauffman equivalence relations is isomorphic to the Kauffman skein module of stacked k-tangles.*

Proof
This follows immediately from the definitions of equivalence by renumbering and Kauffman equivalence. ∎

3.4 Algorithm

In the previous sections we have discussed how we might express a general k-tangle with respect to the Kauffman skein relations. We begin by expressing the k-tangle as a stacked k-tangle multiplied by a braid word from $\mathbb{B}_{2k}$.

We represent stacked k-tangles by k-sequences, and consider multiplying them by the braid word, considered one generator at a time. By Propositions 3.2 and 3.3 we impose conditions to ensure that k-sequences are compatible with a braid generator or inverse; if these are not met then we use the actions of renumbering and rearrangement to express the k-sequence in terms of a linear combination of k-sequences that are compatible.

We provide a rigorous system for making rearrangements and renumberings so that we do not needlessly pass coefficients to and from k-sequences. We want to give as simple a system as possible, and not perform unnecessary operations; the aim of our efforts is to define an algorithm for calculating the Kauffman polynomial of a k-plait, and define it in such a way that it can be implemented without difficulty in a computer language.

The following description of this algorithm follows the flow diagram of Figure 3.10 up to the last decision box (after which we have the concluding part for calculating the Kauffman polynomial of a k-plait).

We consider expressions involving a linear combination of k-sequences. Without loss of generality, take the case when we multiply by a generator σ_i. The process of this algorithm is to ensure that we have a linear combination of k-sequences that are compatible with a generator, in particular that the conditions of Proposition 3.2 (respectively Proposition 3.3 in the case of inverses) are met following a process of rearrangement and renumbering.

In order that we do not perform unnecessary operations, we begin by considering k-sequences that are not compatible with σ_i and have number k in the $(i+1)$th position. By Proposition 3.7 we guarantee that we can perform actions on incompatible k-sequences with number k in the $(i+1)$th position to express them as linear combinations of k-sequences with $k-1$ in that position.

We repeat the procedure for all k-sequences which have $k-1$ in position $i+1$ but which are not compatible, and so on, repeating the process until finally we have performed rearrangements and renumberings for incompatible k-sequences with a 2 in position $i+1$. By reducing numbers in incompatible k-sequences in this way, we remove the possibility of any duplication of work and ensure that we do not miss any incompatible k-sequences.

There is a similar set of steps for the situation that we are multiplying by σ_i^{-1}, in which case we will be concerned with the number in the k-sequence at position i. After completing this series of operations we have a linear combination of k-sequences that are compatible with the generator (or inverse) by Proposition 3.2 (or Proposition 3.3 for an inverse). Due to the way that we ensure compatibility, multiplication involves switching the numbers in the i and $(i+1)$th place in the k-sequences, and multiplying the coefficient of those k-sequences which have the same number in positions i and $i+1$ by v (or v^{-1}).

After this the linear combination of k-sequences is ready to be multiplied by the next generator (or inverse) in the braid word, and so we repeat the process outlined above. When the end of the braid word is reached, the stacked k-tangle multiplied by the braid word will have been expressed as a linear combination of k-sequences (representing stacked k-tangles) in the Kauffman skein.

3.4.1 Calculating the Kauffman polynomial of a k-plait

In order to calculate the Kauffman polynomial of a k-plait we must consider the closure at the bottom of the k-plait structure. Thus, we have to consider how we might calculate the Kauffman polynomial of a stacked k-tangle that is closed off by k cups in the manner for k-plaits. Consider Figure 3.9, where we see the stacked 4-tangle given by (12314234) closed off.

In this case it is not difficult to evaluate the diagram's Kauffman polynomial

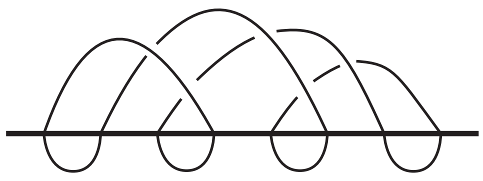


Figure 3.9: Closure of stacked 4-tangle given by 4-sequence (12314234)

(it has value v), but for a sufficiently complicated diagram and large enough k the closure of a stacked k-tangle could be a non-trivial knot, or even a link.

In general we need a different approach. Consider the left-most cup that effects the first two endpoints. If we pull this above the line, and use a Type I Reidemeister move to remove the kink (multiplying by a scalar of v) we see that we now have a stacked 3-tangle with 3-sequence (213123).

Definition

A stacked k-tangle is **closure-compatible** if the introduction of a cup to the two left-most endpoints results in a stacked $(k-1)$-tangle multiplied by a scalar from the set $\{1, v, v^{-1}, \delta\}$.

Definition

A k-sequence, s, is closure-compatible if the stacked k-tangle defined by s is closure-compatible.

As before, we show a condition that ensures closure-compatibility for a k-sequence.

Proposition 3.10 *If the first two numbers of a k-sequence, $s(1)$ and $s(2)$, are such that $|s(1) - s(2)| \leq 1$ then s is closure-compatible.*

Proof

If $s(1) = s(2)$ then the introduction of a cup to the stacked k-tangle represented by s results in a disjoint unknot that we remove by multiplying by δ, leaving a stacked $(k-1)$-tangle that we can represent as a $(k-1)$-sequence.

For the remaining cases we will definitely be able to close off to valid stacked k-tangles, and hence obtain valid k-sequences, as the two arcs are adjacent and thus the closure will not effect the relative ordering of the other arcs. We must consider the pattern that the values of $s(1)$ and $s(2)$ have in the k-sequence. If $|s(1) - s(2)| \leq 1$ then the values of $s(1)$ and $s(2)$ are adjacent, so either $s(1) = a$ and $s(2) = b$ or $s(1) = b$ and $s(2) = b$ for $1 \leq a \leq k-1$ and $b = a+1$.

If $s(1) = a$ and $s(2) = b$ then the two possible patterns in the k-sequence are $(abba)$ and $(abab)$: for the pattern $(abba)$ it is not difficult to see that this will result that closure will result in some $(k-1)$-sequence multiplied by 1 as there is no twisting. For the pattern $(abab)$ we will be able to find a $(k-1)$-sequence after we remove a twist by multiplying by v.

If $s(1) = b$ and $s(2) = b$ then the two possible patterns in the k-sequence are $(baab)$ and $(baba)$: for the pattern $(baab)$ we will once again obtain some $(k-1)$-sequence multiplied by 1. For the pattern $(baba)$ we will obtain a valid $(k-1)$-sequence after removing a twist by multiplying by v^{-1}. ∎

We now discuss k-sequences as being closure-compatible by satisfying the conditions of Proposition 3.10, following the approach that we took previously when we considered conditions ensuring compatibility with braid generators.

As before we perform actions on a linear combination of k-sequences. Again we take advantage of the result of Proposition 3.7 to ensure that our actions proceed in an organised way. In the first instance we act on all k-sequences that have a k in one of the first two positions and which are not closure-compatible. We perform renumbering or rearrangement to reduce k to $k-1$, and perform similar reduction operations in subsequent cycles. This differs from the previous procedure in that the conditions for closure-compatibility are different from the conditions for compatibility. We have to perform fewer cycles through the linear combination of k-sequences, as after we have performed the check for the number 3 we can guarantee that all of the k-sequences will be closure-compatible.

Closing off from k-sequences and $(k-1)$-sequences involves observing where the two numbers lie relative to each other in the k-sequence as explored in

Proposition 3.10. Using these results we can determine the scalar required when we close to a $(k-1)$-sequence.

The simplest way that we can relate the numbers of arcs represented in a k-sequence and the $(k-1)$-sequence that it closes to is by removing the first two numbers in the k-sequence, then subtracting 1 from all of the numbers in the new sequence that are greater than the minimum of the two numbers we removed from the k-sequence. This gives a valid $(k-1)$-sequence, though as before there may not be a unique $(k-1)$-sequence for the stacked $(k-1)$-tangle that is being represented.

We continue in this manner, closing off from linear combinations of m-sequences to linear combinations of $(m-1)$-sequences, until we close off from 2-sequences to the 1-sequence, (11). The coefficient of (11) is the Kauffman polynomial of the closure of the stacked k-tangle (represented as a k-sequence) that we began with.

Hence, by combining this with the algorithm for representing a stacked k-tangle multiplied by a word from the braid group $\mathbb{B}_{2k}$, we can calculate the Kauffman polynomial of a k-plait. We first express the k-plait as a k-sequence multiplied by a braid word, and then use the main algorithm outlined previously to express it as a linear combination of k-sequences. We then close these off using the method described in this section, with the final coefficient of the 1-sequence (11) giving us the Kauffman polynomial of the k-plait.

A flow diagram to illustrate the algorithm can be seen in Figure 3.10.

3.5 Complexity

The number of k-sequences for a fixed k is $\frac{(2k)!}{2^k}$ (Proposition 3.1). Immediately then we can state that the algorithm is not exponential with respect to c, the number of crossings, for a fixed k, in the sense of the general algorithm outlined in Section 1.5.5. Rather than produce an increasing number of diagrams as each crossing is considered, we are able to limit the number of objects by representing stacked k-tangles as k-sequences.

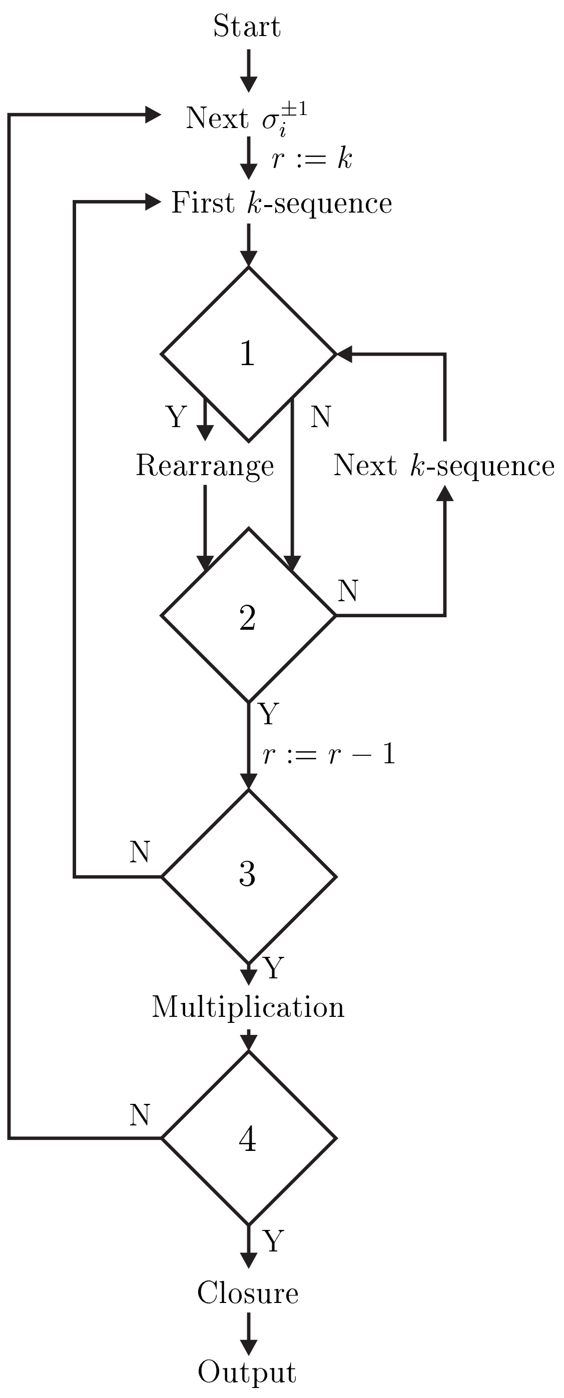


Figure 3.10: Flow diagram showing algorithm for calculating the Kauffman polynomial. At box 1 we look to see if the k-sequence is incompatible with an r at the relevant point; at box 2 we check if we are considering the last k-sequence in the set or not; box 3 checks to see if r is now equal to 1; box 4 checks to see if the end of the input has been reached.

Proposition 3.11 *The number of k-sequences satisfying Proposition 3.2 or Proposition 3.3 for a given generator or inverse $\sigma_i^{\pm 1}$ is*

$$k^2 \frac{(2(k-1))!}{2^{k-1}}.$$

Proof

Consider first of all the case for a generator σ_i. There are $\frac{(2k)!}{2^k k}$ k-sequences with the value r at position i, $1 \leq r \leq k$. A k-sequence s which is not compatible with σ_i is such that $s(i+1) > s(i)$. There are $k-r$ choices for the value of $s(i+1)$, thus we enumerate the number of incompatible k-sequences with r in position i as

$$(k-r)\frac{(2(k-1))!}{2^{k-2}}.$$

Taken over all r we can then evaluate the number of k-sequences compatible with σ_i as

$$\begin{aligned}
\sum_{r=1}^{k}\left(\frac{(2k)!}{2^k k} - (k-r)\frac{(2(k-1))!}{2^{k-2}}\right) &= \frac{(2k)!}{2^k} - \frac{(2(k-1))!}{2^{k-2}}\sum_{r=1}^{k}(k-r) \\
&= \frac{(2k)!}{2^k} - \frac{(2(k-1))!}{2^{k-2}} \times \frac{k(k-1)}{2} \\
&= \frac{(2(k-1))!}{2^{k-1}}\left(\frac{2k(2k-1)}{2} - k(k-1)\right) \\
&= k^2\frac{(2(k-1))!}{2^{k-1}}.
\end{aligned}$$

The same method shows that the value holds for inverses. ∎

From this bound it follows that there is a limit to the number of actions of renumbering and rearrangement required to ensure compatibility. Of course there is growth in terms of the number of operations performed; the number of operations performed to ensure compatibility for the first crossing will be less than the number performed to ensure compatibility for the tenth crossing. For a sufficiently long braid word we will reach a point where our linear combination contains the maximum number of k-sequences. For each successive crossing the number of operations required to ensure compatibility will remain more or less constant, and from this point we could consider the amount of work performed by the main algorithm to be linear with respect to c.

As there is growth in terms of the number of operations performed before this point, we state that the main work of the algorithm is performed in polynomial time, degree 2.

The growth of the coefficients of the k-sequences is not exponential either. $k-1$ passes are made through the set of k-sequences during a sequence of rearrangements, and so at most the spread of z will increase by $k-1$. The spread of v only increases (potentially) during the multiplication stage of the algorithm, and does so by 1 at most as we multiply relevant k-sequences by v or v^{-1} depending on whether we multiply by a generator or an inverse. Hence, for coefficients we have linear growth in z and linear growth in v, giving quadratic growth overall in coefficients.

As the action of the main algorithm is polynomial, degree 2, and the growth of coefficients is polynomial degree 2 (all with respect to c for a fixed k) then the algorithm as a whole is a polynomial time algorithm of degree 4. However, once the number of k-sequences reaches a certain bound the number being acted on by subsequent will be constant with only minor fluctuations. As the number of operations performed is bounded the main algorithm becomes linear with respect to c. Hence the overall algorithm will effectively, from that point on, be polynomial degree 3 with respect to c for a fixed k.

3.6 Implementation

The algorithm described in this chapter lends itself to implementation in a computing language. Although the algorithm works in polynomial time, as with other algorithms for calculating polynomial invariants of knots it is too complicated to allow any serious calculation by hand.

In Appendix A.2 we give a listing for a Maple procedure that implements the algorithm developed in this chapter. The code is documented in that appendix, but there are a few points that are worth touching on here.

The first is to note that we use the "`permute`" command in Maple to generate the set of k-sequences at the start of a calculation. We use this command on the numbers in the string equivalent to the k-sequence $(1122\ldots kk)$, which has the effect of creating a list with a very regular ordering that we can exploit.

The actions of rearrangement and renumbering are performed by looking in relevant places within k-sequences, and seeing how adjacent numbers lie relative to each other. By considering a few simple cases and factors we determine the appropriate course to take, i.e., rearrangement or renumbering.

There are two main arrays of information kept in memory by the program. One is the array holding the set of k-sequences, and the other is the table of coefficients that are paired with the k-sequences. The actions of rearrangement, renumbering and multiplication are performed by altering coefficients in the second array to reflect changes in the linear combination of k-sequences.

I have not found a command in Maple that simply gives the index of a k-sequence in the first array, so I created a simple routine that allows us to narrow the list of entries that we search through. `SeqIndex` is listed in Appendix A.1, and exploits the regular ordering that the `permute` command gives the list of k-sequences.

3.7 Plait Presentations

In Appendix B we give tables of plait presentations of all knots up to ten crossings; while plait presentations are fairly well known (as "$2k$-plats") it seems that there is no record of the braid words for plait presentations. These tables record presentations for all knots up to ten crossings. The source diagrams are due to the Rolfsen Knot Table as recorded at the Knot Atlas [52], with some additional diagrams from KnotInfo [26].

While plait presentations of knot diagrams are not in principle difficult to obtain, it can be difficult to find presentations of minimal width; of course, a plait presentation might not have the minimal number of crossings for the link presented. All of the presentations that I give have minimal width, as each has a width equal to the knot's bridge index, but it remains to be seen whether some of them could have the number of crossings improved, i.e., by reducing the length of the braid word that we close-off in the plait format.

3.8 Discussion

There are some questions that the work of this chapter raises; there are possible extensions that we can make to the theory as well, but we will examine those in detail in the following chapter.

3.8.1 The number of stacked k-tangles

There are $\frac{(2k)!}{2^k}$ k-sequences, and the implementation of the algorithm that I have given for this operates on a spanning set which is the entire set of k-sequences. There are lots of interesting questions that we can ask regarding spanning sets for the space of k-sequences and for stacked k-tangles.

For example, we can begin by giving the size of a basis for the k-sequences in the Kauffman skein.

Lemma 3.12 *A basis for the space of k-sequences contains $\prod_{r=1}^{k}(2r-1)$ elements.*

Proof
Let S be the set of k-sequences such that the numbers in a k-sequence in S occur in counting order as the sequences are examined from left to right.

We can perform renumbering and rearrangement operations on any k-sequence to express it as a linear combination of k-sequences from the set S by Proposition 3.7. The set S is thus a spanning set of the space of k-sequences.

We calculate the size of S as follows: there is a 1 in the first place of the sequence, and there are $2k-1$ possible places where the other 1 could be. By removing these we have a sequence of length $2k-2$ with a 2 at the start; there are $2k-3$ places that we can place the other 2. We continue in this way, and see that the number of possible sequences of this format is the product of all of the odd numbers from 1 to $2k-1$, or

$$|S| = \prod_{r=1}^{k}(2r-1).$$

We cannot remove any element of S and express it as a linear combination of the others, hence the elements of S form a basis of the space of k-sequences. ∎

A problem that I have not been able to answer is the question of how many stacked k-tangles there are for a given k. For $k = 2$ and $k = 3$ there are few enough k-sequences that we can enumerate the set of stacked k-tangles by inspection. For $k = 4$ we have 2520 diagrams to consider and the task becomes too difficult to consider simply by comparing all of the stacked 4-tangle diagrams.

By considering relative positions of neighbouring arcs in the set of 4-sequences we can eliminate duplicate sequences that represent stacked 4-tangles which do not have unique numberings. This argument, however, does not guarantee that the 4-sequences it obtains account for all of the duplicate copies of stacked 4-tangles, and it unfortunately also produces "false positives." In the absence of further results, the quoted number of 550 must remain an upper bound, with a lower bound provided by the number of stacked 4-tangles in a basis. We display these with the values for $k = 2$ and $k = 3$ in Table 3.1.

k	$\lvert\{\text{Stacked } k\text{-tangles}\}\rvert$
2	4
3	35
4	$x, 105 < x \leq 550$

Table 3.1: Size of sets of stacked k-tangles

It is possible that a combinatorial answer exists in [10]; however, to date, I have not been able to understand all of the terminology and results in the paper in order to fully decide whether an answer to the problem exists there.

Now that the upper bound has been reduced from 2520 to 550 it is possible that the remaining work has been reduced enough to confirm by inspection the size of the set of stacked 4-tangles. However, this approach will be too time-consuming for calculating the size of the set of stacked k-tangles in general.

3.8.2 Improving the algorithm

The natural question that one might ask is whether or not an even better, more efficient algorithm exists for calculating the Kauffman polynomial. While this question is much too broad to answer in general, there are some points that we can note for the case of calculating the Kauffman polynomial of a k-plait.

In Chapter 6 we show results for repeated twists on two strings in a braid and how these can be expressed in the Kauffman skein. These results may have some application here, either in simplifying diagrams before a run of the algorithm or by supplementing the algorithm.

One thought that I have examined is the possibility of working from a basis of the space of k-sequences. While it is true that we can express any linear combination of k-sequences as a linear combination of basis elements it does not follow that these basis elements will be compatible with a given generator or inverse. Extending this idea, we might consider working with two bases, and rearranging from expressions in one basis to another to obtain compatibility; however, a little experimentation shows that two bases will not be enough to ensure compatibility, and combined with the extra operations that an implementation would be required to perform it is not clear that we would be reducing the work performed.

In the extensions section of the next chapter we discuss how one might calculate a 2-parallel of a k-plait, and how we can use the methods of this chapter in order to reduce the work needed in those cases. These are based around situations where the braid word gives a long arc crossing over (or under) other braid strings. In this case we perform rearrangements to allow the arc to cross over (or under) all of these strings in the act of one multiplication, rather than in several stages. This affords a reduction of the amount of work done in comparison.

3.8.3 Numbering arcs in layers

One problem with the algorithm that we have outlined is that it performs operations to ensure compatibility (Propositions 3.2 and 3.3), rather than simply performing operations on k-sequences that are incompatible.

Consider the example of multiplying (11442233) by σ_2, which is not incompatible. The algorithm that we have outlined would tell us to use renumbering actions to rearrange 4 and 3, then 3 and 2, and then 2 and 1 before the multiplication could be performed. In this case, all of the arcs in the stacked 4-tangle are in the same layer, and so the numbering that we give them is arbitrary in some respect.

An improvement to the algorithm would be to consider the numbering of arcs which are in the same layer as being irrelevant. The difficulty with this approach is that essentially we are considering the stacked k-tangle diagrams rather than representations. The machinery of any such implementation would doubtless be increased dramatically to allow for these possibilities.

Chapter 4

Stacked k-tangles and the Homfly Polynomial

4.1 Introduction

We extend the definition of stacked k-tangles to allow oriented arcs, and subsequently develop a method for calculating polynomial invariants for oriented links; the example that we give is for calculating the Homfly polynomial of links given as plait presentations. The steps leading to this algorithm are similar to the work of the previous chapter, and so we state many of the results without proof. These lead to Proposition 4.8 which is the key result for the algorithm.

As with the previous algorithm for calculating the Kauffman polynomial for knots presented as plaits, we show that this algorithm is a polynomial time algorithm for a fixed k with respect to the number of crossings c. There are existing polynomial time algorithms for calculating the Homfly polynomial, but these are based on braid presentations [45]. After searching through the literature I believe that the algorithm given in this chapter is the first polynomial time algorithm not based around presentations for closed braids.

We conclude the chapter by considering some extensions to the general theory of calculating polynomial invariants by representing stacked k-tangles as k-sequences (oriented or unoriented). We investigate several possibilities for reducing the amount of work that our existing algorithm performs. We consider this with respect to calculating the Homfly polynomial of a reverse parallel of a knot; this generalises to m-parallels in general (for Homfly and Kauffman). We briefly consider a possible application by shifting from words in a braid group to considering words written in terms of band generators [8], [23].

4.2 Oriented stacked k-tangles

As the Homfly polynomial is an invariant for oriented links, we must now consider tangles with oriented arcs. We use the k-tangle from the previous chapter as our starting point.

Definition

An **oriented k-tangle** is a k-tangle with each arc oriented.

See Figure 4.1 for examples of oriented k-tangles.

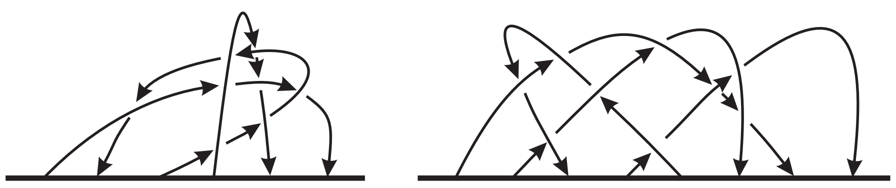


Figure 4.1: Examples of oriented 3- and 4-tangles

Definition

An **oriented stacked k-tangle** is a stacked k-tangle with each arc oriented.

In Figure 4.2 we see two examples of oriented stacked 4-tangles; these are the two examples from the previous chapter with arcs now oriented.

We give a numbering to the arcs of oriented stacked k-tangles in exactly the same way as we did previously for unoriented stacked k-tangles. Diagrammatically we see numbered arcs with the orientation indicated on the arcs (as in the example of Figure 4.3, showing a numbering for the left-hand example of

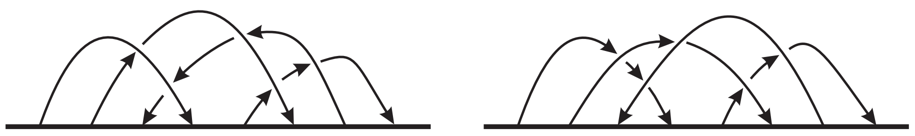


Figure 4.2: Examples of oriented stacked 4-tangles

Figure 4.2). The orientation is extra information that we have to pass to the analogue of k-sequences for this oriented case.

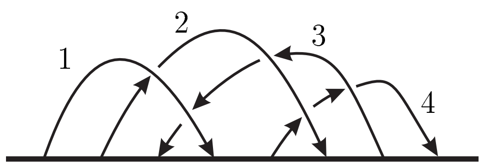


Figure 4.3: An oriented stacked 4-tangle with numbered arcs

Definition

An **oriented k-sequence** is a sequence of numbers, $\pm i, 1 \leq i \leq k$, representing the endpoints of arcs of an oriented stacked k-tangle. The absolute value of the number indicates the arc and we take the convention that the orientation of the arc runs in the direction from $-i$ to $+i$.

For example the oriented stacked 4-tangle given in Figure 4.3 has oriented 4-sequence $(^{-}1\,^{-}2\,3\,1\,^{-}4\,2\,^{-}3\,4)$. As in the previous chapter, the set of oriented k-sequences is larger than the set of oriented k-tangles.

Proposition 4.1 *The set of oriented k-sequences has $(2k)!$ elements.*

Proof

This follows immediately as we permute $2k$ different numbers. ∎

4.3 Action of braid generators

In Section 3.3 of the previous chapter we showed that it was possible to express a k-sequence as a linear combination of k-sequences satisfying certain conditions. We did this specifically with the aim of showing that one could express a stacked k-tangle composed with a braid word from $\mathbb{B}_{2k}$ as a linear combination of stacked k-tangles. In this section we show that similar results can be obtained for Homfly.

We begin by returning to compatibility, defining the concept for oriented stacked k-tangles and oriented k-sequences.

Definition

An oriented stacked k-tangle t_1 is **compatible** with a generator or inverse $\sigma_i^{\pm 1}$ if the result of multiplying t_1 by $\sigma_i^{\pm 1}$ gives another oriented stacked k-tangle t_2, or t_1 multiplied by a scalar $v^{\pm 1}$.

Definition

An oriented k-sequence s is **compatible** with a generator or inverse $\sigma_i^{\pm 1}$ if the oriented stacked k-tangle determined by s is compatible with $\sigma_i^{\pm 1}$.

As orientation is kept *with the arcs* in the tangle, we can see that orientation does not have a direct bearing on compatibility. We can impose the following conditions to ensure compatibility as we did with Propositions 3.2 and 3.3 in the previous chapter.

Proposition 4.2 *If the absolute value of the number at position i in an oriented k-sequence s is greater than or equal to the absolute value of the number at position $i+1$ then s is compatible with σ_i.*

Proposition 4.3 *If the absolute value of the number at position i in an oriented k-sequence s is less than or equal to the absolute value of the number at position $i+1$ then s is compatible with σ_i^{-1}.*

The proofs of Propositions 4.2 and 4.3 are essentially the same to those of Propositions 3.2 and 3.3.

We now move on to show that similar concepts of renumbering and rearrangement can be applied in the Homfly skein module of oriented k-sequences, respecting operations in the Homfly skein module of oriented stacked k-tangles.

4.3.1 Renumbering

Definition

Let s and s' be oriented k-sequences. We say that s is *equivalent by renumbering* to s' if and only if s' arises from a valid numbering for the same oriented stacked k-tangle that s determines.

The following two propositions mirror the propositions related to renumbering in the previous chapter, Propositions 3.4 and 3.5. There are four parts to each, owing to the four different possibilities for orientation in oriented stacked 2-tangles. We state them without proof.

Proposition 4.4 *Let $1 \leq a \leq k-1$ and $b = a+1$.*

1. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = {}^-a & s_1(r) = b & s_1(t) = {}^-b \\ s_2(p) = b & s_2(q) = {}^-b & s_2(r) = a & s_2(t) = {}^-a \end{array}$$

   and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

   Then s_1 is related to s_2 by renumbering.

2. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = {}^-a & s_1(r) = {}^-b & s_1(t) = b \\ s_2(p) = b & s_2(q) = {}^-b & s_2(r) = {}^-a & s_2(t) = a \end{array}$$

   and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

   Then s_1 is related to s_2 by renumbering.

3. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = {}^-a & s_1(q) = a & s_1(r) = b & s_1(t) = {}^-b \\ s_2(p) = {}^-b & s_2(q) = b & s_2(r) = a & s_2(t) = {}^-a \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

4. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = {}^-a & s_1(q) = a & s_1(r) = {}^-b & s_1(t) = b \\ s_2(p) = {}^-b & s_2(q) = b & s_2(r) = {}^-a & s_2(t) = a \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

Proposition 4.5 *Let $1 \leq a \leq k-1$ and $b = a+1$.*

1. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = b & s_1(r) = {}^-b & s_1(t) = {}^-a \\ s_2(p) = b & s_2(q) = a & s_2(r) = {}^-a & s_2(t) = {}^-b \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

2. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = {}^-b & s_1(r) = b & s_1(t) = {}^-a \\ s_2(p) = b & s_2(q) = {}^-a & s_2(r) = a & s_2(t) = {}^-b \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

3. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = {}^-a & s_1(q) = b & s_1(r) = {}^-b & s_1(t) = a \\ s_2(p) = {}^-b & s_2(q) = a & s_2(r) = {}^-a & s_2(t) = b \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

4. *Consider two oriented k-sequences s_1 and s_2, such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = {}^-a & s_1(q) = {}^-b & s_1(r) = b & s_1(t) = a \\ s_2(p) = {}^-b & s_2(q) = {}^-a & s_2(r) = a & s_2(t) = b \end{array}$$

and $s_1(i) = s_2(i)$ for all other $1 \leq i \leq 2k$.

Then s_1 is related to s_2 by renumbering.

4.3.2 Rearrangement

Definition

An oriented k-sequence s is *Homfly equivalent* to $\sum a_i s_i$, a linear combination of oriented k-sequences with coefficients from the Homfly skein module, if and only if a linear combination of oriented stacked k-tangles corresponding to the linear combination of oriented k-sequences can be obtained from the oriented stacked k-tangle determined by s by applying the Homfly skein relation.

We say that we use a *rearrangement* action when applying Homfly skein relations in order to obtain a Homfly equivalent linear combination of oriented k-sequences.

It follows that two oriented k-sequences that are equivalent by renumbering are Homfly equivalent.

Whereas in the case of Kauffman equivalence we had one relation that we showed for adjacent arcs, in the case of Homfly equivalence there are four relations that we must make clear. We state them in the next two propositions, which are proved in a similar way to the proof of Proposition 3.6.

Proposition 4.6 *Let $1 \leq a \leq k-1$ and $b = a+1$.*

1. *Consider three oriented k-sequences s_1, s_2, s_3 such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = {}^-b & s_1(r) = {}^-a & s_1(t) = b \\ s_2(p) = b & s_2(q) = {}^-a & s_2(r) = {}^-b & s_2(t) = a \\ s_3(p) = a & s_3(q) = {}^-a & s_3(r) = {}^-b & s_3(t) = b \end{array}$$

   and for all other $1 \leq i \leq 2k$ $s_1(i) = s_2(i) = s_3(i)$.

   Then s_1 is Homfly equivalent to $s_2 - zs_3$.

2. *Consider three oriented k-sequences s_1, s_2, s_3 such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = {}^-a & s_1(q) = b & s_1(r) = a & s_1(t) = {}^-b \\ s_2(p) = {}^-b & s_2(q) = a & s_2(r) = b & s_2(t) = {}^-a \\ s_3(p) = {}^-a & s_3(q) = a & s_3(r) = b & s_3(t) = {}^-b \end{array}$$

   and for all other $1 \leq i \leq 2k$ $s_1(i) = s_2(i) = s_3(i)$.

   Then s_1 is Homfly equivalent to $s_2 - zs_3$.

Proposition 4.7 *Let $1 \leq a \leq k-1$ and $b = a+1$.*

1. *Consider three oriented k-sequences s_1, s_2, s_3 such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = a & s_1(q) = b & s_1(r) = {}^-a & s_1(t) = {}^-b \\ s_2(p) = b & s_2(q) = a & s_2(r) = {}^-b & s_2(t) = {}^-a \\ s_3(p) = a & s_3(q) = b & s_3(r) = {}^-b & s_3(t) = {}^-a \end{array}$$

   and for all other $1 \leq i \leq 2k$ $s_1(i) = s_2(i) = s_3(i)$.

   Then s_1 is Homfly equivalent to $s_2 + zs_3$.

2. *Consider three oriented k-sequences s_1, s_2, s_3 such that for $1 \leq p < q < r < t \leq 2k$*

$$\begin{array}{llll} s_1(p) = {}^-a & s_1(q) = {}^-b & s_1(r) = a & s_1(t) = b \\ s_2(p) = {}^-b & s_2(q) = {}^-a & s_2(r) = b & s_2(t) = a \\ s_3(p) = {}^-a & s_3(q) = {}^-b & s_3(r) = b & s_3(t) = a \end{array}$$

and for all other $1 \leq i \leq 2k$ $s_1(i) = s_2(i) = s_3(i)$.

Then s_1 is Homfly equivalent to $s_2 + zs_3$.

Having given these statements we are in a position to give a result that mirrors Proposition 3.7, which was the cornerstone of the algorithm that we outlined for calculating the Kauffman polynomial of k-plaits.

Proposition 4.8 *An oriented k-sequence s with number r at position j where $2 \leq r \leq k$ is Homfly equivalent to a linear combination of oriented k-sequences each with $r-1$ at position j.*

An oriented k-sequence s with number $-r$ at position j where $2 \leq r \leq k$ is Homfly equivalent to a linear combination of oriented k-sequences each with $-(r-1)$ at position j.

Proof
This follows from Propositions 4.4 - 4.7 by similar considerations to the proof of Proposition 3.7. ∎

Proposition 4.9 *An oriented k-sequence s that is incompatible with a generator or inverse $\sigma_i^{\pm 1}$ is Homfly equivalent to a linear combination of oriented k-sequences compatible with $\sigma_i^{\pm 1}$.*

Proof
This follows from Proposition 4.8. ∎

4.4 Algorithm

In this section we outline the ways in which this algorithm differs from that of the previous chapter. Most of these considerations are due to how we make allowances for dealing with the orientation information encoded in oriented k-sequences.

We begin with a k-plait presentation represented as a stacked k-tangle multiplied by a braid word from $\mathbb{B}_{2k}$ closed off by k cups at the bottom of the presentation. We assign an orientation, or orientations if dealing with a link, and determine the initial orientations of the arcs in the stacked k-tangle.

Proposition 4.8 shows that the signs of numbers in oriented k-sequences do not change as we perform operations. As a result, we can use the set of (unoriented) k-sequences along with one sequence to record the orientations of the arcs that the k-sequences represent. In this way the algorithm operates considering a much smaller set of objects: we return to considering the $\frac{(2k)!}{2^k}$ k-sequences plus a sequence of 1s and -1s that contain the information for the orientation of arcs.

The previous algorithm for calculating the Kauffman polynomial worked in two stages: first we performed a series of renumberings and rearrangements in order to ensure that k-sequences were compatible with the next generator in the braid word. Then we multiplied the k-sequences in our linear combination, essentially switching the two numbers at the appropriate point in the k-sequences or multiplying coefficients by $v^{\pm 1}$ if the endpoints represented belonged to the same arc.

Thus the algorithm for calculating the Homfly polynomial of a knot presented as a plait presentation functions in the same way as that for the Kauffman polynomial: we perform operations on k-sequences, rearranging the linear combination at each stage so that all of the k-sequences are compatible with the next generator. The rearrangements are decided by how adjacent-numbered arcs are related in the k-sequence and from the sequence of 1s and -1s that carry the orientation information.

As with the algorithm of Chapter 3, we proceed at each stage by ensuring that generators and inverses satisfy the compatibility conditions of Propositions 4.2 and 4.3. We consider k-sequences with the number k in the affected position, and work to reduce this number by renumbering or rearrangement to $k-1$ if the conditions for compatibility are not met; we then work in turn on k-sequences with $k-1$ in that position and so on. Renumbering is the same as before, as that operation on the oriented k-sequence reflects the fact that two arcs are in the same layer in the oriented stacked k-tangle.

Rearrangement in the Homfly case is not as straight-forward as the Kauffman case as we have additional information given by the orientation. The orientation of adjacent arcs has a bearing on the application of the skein relations, particularly the oriented k-sequence representing the smoothing. While this is extra information to consider in our application of the algorithm, it is not something that is extremely difficult to resolve, and there are only a very limited number of cases to be considered. The relationships for all of these can be seen in Propositions 4.6 and 4.7.

Once we have completed a series of renumbering and rearrangements we have a linear combination of oriented k-sequences (by combining the k-sequences and the information of the sequence of signs) that are compatible with the required generator or inverse; we multiply and then move on to the next generator or inverse. In this way we express an oriented k-sequence composed with a braid word from $\mathbb{B}_{2k}$ as a linear combination of oriented k-sequences. In calculating the Homfly polynomial of a k-plait these actions take us to the point of considering closure by k cups much as it did in the case of the algorithm for calculating the Kauffman polynomial.

The action of closing off proceeds in the same manner as for the algorithm of the previous chapter. We will not discuss this in detail here as the procedure is so similar: we perform rearrangements and renumberings on the linear combination of k-sequences (with information from the sequence of signs) to satisfy an analogous condition to Proposition 3.10 ensuring closure-compatibility.

4.5 Complexity

The algorithm outlined in this chapter differs from the algorithm of the previous chapter, but only in the respect that a rearrangement operation now expresses a k-sequence as a linear combination of two other k-sequences, whereas in the algorithm for calculating the Kauffman polynomial a rearrangement action expressed it as a linear combination of three k-sequences. This does not change the order of complexity of the algorithm.

The size of coefficients in v and z grow quadratically with respect to c as in the previous algorithm. Hence, considered together, the algorithm works in polynomial time, degree 4, with respect to c for a fixed k.

As with the algorithm for the Kauffman polynomial, after a certain point the algorithm will essentially perform the same amount of work with each subsequent crossing. From this point we can view the algorithm as a whole as being polynomial degree 3. Each generator in the braid word after the critical point has been reached will act on a set of roughly the same size. It is possible that terms can combine and reduce the number of k-sequences in an expression, but it will not vary greatly.

The main area that the complexity necessarily differs in is the fact that rearrangement in the algorithm for Homfly is expressed in terms of two k-sequences rather than three. There are fewer terms in a rearrangement operation and so the growth of the number of terms in the linear combination of k-sequences is less rapid. As we use the k-sequences plus a sequence of signs the number of k-sequences compatible with a given generator or inverse will be the same as in the case for the Kauffman algorithm under the conditions of Proposition 3.11.

4.6 Implementation

In Appendix A.3 we give the listing of the code for this algorithm, implemented once again in Maple; it is well documented and commented, and so we will now briefly consider the few areas where it deviates from the algorithm for Kauffman in Appendix A.2.

As before we use the "`permute`" command to obtain the full set of k-sequences. If we were to consider the explicit set of oriented k-sequences we could generate them in the same way. The problem with using the set of oriented k-sequences to keep track of coefficients is that there are substantially more oriented k-sequences than there are (unoriented) k-sequences. We are fortunate that we have the observation about the signs of endpoints so that we can use the set of k-sequences plus one other sequence which stores the information about the signs of endpoints. This drastically reduces the number of elements that we must keep in memory and search through.

Once again we use the subroutine `SeqIndex` (Appendix A.1) in order to obtain the index of a k-sequence that we require, either for renumbering, rearrangement or multiplication. In the absence of a direct command which could take us to a desired k-sequence this is a useful routine to have.

The only advantage we would have in using the oriented k-sequences is that we could have extended `SeqIndex` to obtain the index of a desired k-sequence: as any oriented k-sequence is a permutation of $2k$ distinct symbols, and given that we know how Maple permutes elements in a list, we can derive a system for finding one of these elements.

The implementation that we give operates under the assumption that the order of signs in the starting sequence is $(-1, 1, -1, 1, \ldots, -1, 1)$. This is easy enough to force using Type I Reidemeister moves, but if this is inconvenient then the program could be easily altered so that it takes the starting configuration of the sequence of signs as another argument.

4.7 Discussion

In this section we discuss ways in which the work of the last two chapters can be extended, either to look at problems that arise from the theory we have discussed or to look at ways in which we can improve on what I have outlined.

4.7.1 Reverse parallel satellites

In Chapter 2 we considered an extension to the result of Rudolph regarding Homfly polynomials of reverse parallels of knots [54]. When we consider the reverse parallel of a k-plait with c crossings, we are essentially considering a plait of width $2k$ (although the closure is not immediately that of a plait as we have defined it) with $4c$ crossings.

A 2-parallel of a braid word from $\mathbb{B}_{2k}$ is a word from the braid group $\mathbb{B}_{4k}$, and words are mapped by their generators according to the following map:

$$\phi : \sigma_i^{\pm 1} \to \sigma_{2i}^{\pm 1}\sigma_{2i+1}^{\pm 1}\sigma_{2i-1}^{\pm 1}\sigma_{2i}^{\pm 1}.$$

We move from considering linear combinations of k-sequences to linear combinations of $2k$-sequences. This dramatically increases both the number of sequences considered and the number of sequences that will be compatible with a particular generator or inverse. Given that we will be considering four times as many crossings, we need to do everything that we can in order to reduce the amount of work performed by the algorithm.

We consider the action of multiplication and conditions that ensure compatibility in order to reduce the amount of work performed by the algorithm. Consider the diagram of a 2-parallel of generator σ_i in Figure 4.4.

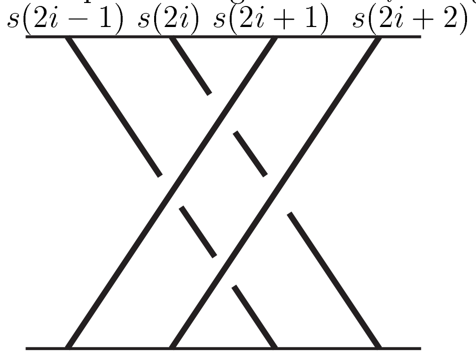


Figure 4.4: 2-parallel of generator σ_i, $\sigma_{2i}\sigma_{2i+1}\sigma_{2i-1}\sigma_{2i}$

Proposition 4.10 *A $2k$-sequence s with $s(2i-1) = s(2i)$ or $s(2i+1) = s(2i+2)$ is compatible with $\sigma_{2i}\sigma_{2i+1}\sigma_{2i-1}\sigma_{2i}$, the 2-parallel of a generator $\sigma_i \in \mathbb{B}_{2k}$.*

Proof
The stacked $2k$-tangle determined by the $2k$-sequence in this case would have an arc joining one of the two possible adjacent positions, and the four generators in the 2-parallel of the single crossing would switch the location of the joined area. See Figure 4.5 for an illustration. ∎

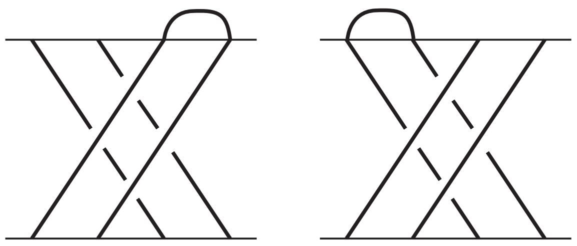


Figure 4.5: Exceptional compatibility

If the conditions of Proposition 4.10 are met then we say that the $2k$-sequence has **exceptional compatibility** with the 2-parallel of σ_i.

Proposition 4.11 *If a $2k$-sequence s is such that*

$$s(2i+1) \leq \ min\{s(2i-1), s(2i)\} \text{ and } s(2i+2) \leq \ min\{s(2i-1), s(2i)\}$$

then s is compatible with $\sigma_{2i}\sigma_{2i+1}\sigma_{2i-1}\sigma_{2i}$, the 2-parallel of a generator $\sigma_i \in \mathbb{B}_{2k}$.

Proof
In order to satisfy Proposition 4.2 it must be the case that $s(2i+1) \leq s(2i-1)$ and $s(2i+1) \leq s(2i)$, and also $s(2i+2) \leq s(2i-1)$ and $s(2i+2) \leq s(2i)$, as these reflect overcrossing arcs in the stacked $2k$-tangle. However, it cannot be true that both $s(2i+1) = s(2i-1)$ and $s(2i+1) = s(2i)$ (and similarly for $s(2i+2)$). Hence to satisfy compatibility conditions

$$s(2i+1) \leq \ \min\{s(2i-1), s(2i)\} \text{ and } s(2i+2) \leq \ \min\{s(2i-1), s(2i)\}.$$

∎

Our approach then is to use renumbering and rearrangement as before so that these conditions are satisfied and compatibility is ensured. In the approach we can exclude any $2k$-sequences that satisfy exceptional compatibility, and focus on those that still need attention.

In the following discussion we refer to the 2-parallel of a braid generator; similar statements can be made for the 2-parallel of an inverse.

As we need both $s(2i+1)$ and $s(2i+2)$ to be less than or equal to the minimum of $\{s(2i-1), s(2i)\}$ it makes sense as a first step to perform initial rearrangements and renumberings on values $s(2i-1)$, $s(2i)$; the nature of these operations is to perform two passes, the first of which acts to increase an occurrence of a 1 to a 2, and then to increase an occurrence of a 2 to a 3 in either of $s(2i-1)$ or $s(2i)$. This guarantees that neither $s(2i-1)$ or $s(2i)$ take the minimum value.

We perform the usual cycle of renumbering and rearrangement on the value of endpoint $s(2i+1)$. In this case we are performing these operations only to the point that $s(2i+1) \leq \min\{s(2i-1), s(2i)\}$.

Upon completion of this series of operations, we act on the endpoint $s(2i+2)$, and repeat the cycle of operations so that $s(2i+2) \leq \min\{s(2i-1), s(2i)\}$. When this is satisfied for all $2k$-sequences in the linear combination we have an expression that is compatible with the 2-parallel of a single crossing, and we perform multiplication in the usual way (at the $2k$-sequence level, by moving numbers in the sequence and multiplying by v if necessary).

For $2k$-sequences that have been involved in rearrangements and renumberings to ensure compatibility for $s(2i+1)$ it can be seen that less work is performed to then ensure compatibility for $s(2i+2)$ also. At most we perform two full cycles of rearrangement and renumbering, and increase some values of $s(2i-1)$ and $s(2i)$. Considering the situation of Figure 4.4 in the usual manner would involve performing four cycles of rearrangements and renumberings to ensure compatibility, as well as intermediate multiplication steps.

Effectively we have halved the amount of work done in terms of the number of operations performed than if we had simply considered this as a $2k$-plait with $4c$ crossings. Given that the set of $2k$-sequences is much larger than the set of k-sequences the bound on the number of operations that have to be performed in order to ensure regular compatibility is much larger; but we still have a saving in the amount of work that must be done in order to perform multiplication. This equates to roughly the amount of work done in calculating the polynomial for a $2c$-crossing $2k$-plait with the normal closure, as opposed to this $4c$-crossing $2k$-plait that has a "doubled" closure.

If the initial sign sequence for the k-plait presentation is one of alternating $+1$s and -1s then the sign sequence for the reverse parallel will also be alternating $+1$s and -1s. As multiplication by the four generators from the doubling up of a single crossing swaps two pairs of numbers, we can easily see that the sign sequence will remain constant throughout the operation of the algorithm. The sign sequence for the reverse parallel can be recovered by considering the position of an endpoint in the sequence: endpoints in positions $2n, 1 \leq n \leq 2k$ have sign $+1$, while endpoints in positions $2n-1, 1 \leq n \leq 2k$ have sign -1.

This example was motivated by an example for the Homfly polynomial, but the principle of reducing the work of the main algorithm applies equally to calculating the Kauffman polynomial of 2-parallels.

4.7.2 Band-generators

Another possible extension to the general principle is to consider the case of band-generator style presentations ([8] and [23]).

A generator a_{ts}, in band-generator notation, reflects a potentially long word in Artin braid presentations, with

$$a_{ts} = (\sigma_{t-1}\sigma_{t-2}\ldots\sigma_{s+1})\sigma_s(\sigma_{s+1}^{-1}\ldots\sigma_{t-2}^{-1}\sigma_{t-1}^{-1})$$

for $1 \leq s < t \leq 2k-1$ when taken from the braid group $\mathbb{B}_{2k}$. The feature that we are interested in are the parts of the band-generator in standard braid notation of the form $\sigma_r\sigma_{r-1}\sigma_{r-2}\ldots\sigma_{r-a}$, i.e., one string crossing over many strings.

Rearrangements and renumberings could be performed to ensure that the linear combination of k-sequences is compatible with the word $\sigma_r\sigma_{r-1}\sigma_{r-2}\ldots\sigma_{r-a}$, rather than by considering each generator in turn. As with the reverse parallel case, we perform rearrangements and renumberings so that $s(r+1) \leq \min\{s(r), s(r-1), \ldots, s(r-a)\}$; there will also be conditions for k-sequences with exceptional compatibility, in a similar manner to how it was considered previously.

While I have not examined this idea in detail, I believe that there are interesting questions that could be explored at a later date. The main question that could be explored is whether band-generator presentations for knots can be used in conjunction with the approach that I have outlined for calculating polynomial invariants, in order to reduce the work performed by the algorithm.

A more technical question is whether an implementation (in some programming language) as we have previously described it could benefit from noticing sequences of generators such as $\sigma_r\sigma_{r-1}\sigma_{r-2}\ldots\sigma_{r-a}$, and whether this would then allow a saving in work performed and calculation time rather than considering each of the generators in turn.

4.7.3 Subsets of k-sequences

The size of the set of k-sequences grows drastically as k grows. As noted in Chapter 1 it might often be easier to obtain a plait presentation with width greater than the bridge index. However, any calculations using the algorithms that we have outlined would be performing operations on a large set of objects; for $k = 6$ there are over seven million 6-sequences to consider.

One strategy might be to begin with the starting sequence, $(1122\ldots kk)$, and from that generate the k-sequences that are in use, i.e., those with non-zero coefficients. In this way we restrict ourselves to only having a subset of the k-sequences (and any coefficients) in memory; for presentations that are *wide and short*, i.e., with relatively large k and small number of crossings c, this could be an asset in allowing computation when generation and management of the entire set of k-sequences in memory would be impractical.

Of course, this strategy would not be practical in general for large values of c as the growth of the number of k-sequences being stored might be too rapid to allow calculation. Also, we would not be able to optimise the search routines for the operations requiring us to move coefficients unless we added yet more structure and procedures to an implementation to order the k-sequences in memory.

4.7.4 Morse link presentations

Consider the two diagrams in Figure 4.6. The diagram on the left is a 4-plait

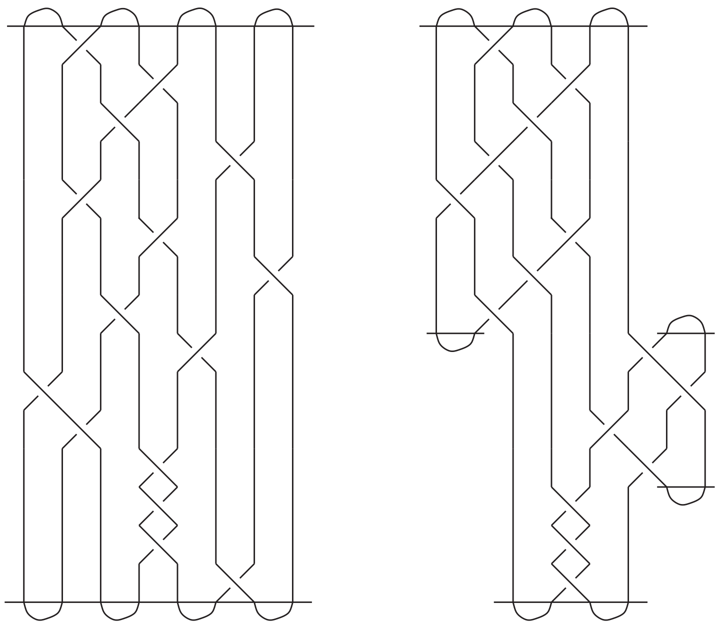


Figure 4.6: Presentations of the Kinoshita-Teresaka knot

presentation of the Kinoshita-Teresaka knot; the diagram on the right shows the same presentation altered to show one important feature. The original plait presentation given has width 4, but the right-hand diagram has width 3 for the most part; we close off one cup (to the left-most strings) and introduce another cap and strings on the opposite side of the presentation, and continue with the rest of the presentation as width 3, then width 2.

While the presentation on the right of Figure 4.6 is not strictly a plait presentation it does offer advantages for calculation if we consider our methods.

Calculating a polynomial invariant of a 4-plait involves performing operations on the set of 4-sequences, which has 2520 elements. The set of 3-sequences has only 90 elements, and these are all that we would need to consider for the first half of the braid word. We could then close off and pass coefficients to an appropriate linear combination of 3-sequences as we introduce another cap.

While it is true that there is a fixed finite number of operations required to calculate a polynomial invariant for a 4-plait with c crossings, the corresponding number for a 3-plait with c crossings will be much smaller. The implementations that we have outlined operate by performing cycles of operations on the set of k-sequences, and must cycle through the entire set $k-1$ times in order to check conditions for compatibility. If instead we are able to act on the set of $(k-1)$-sequences we are acting in a much smaller set of elements, and we also have to perform fewer cycles.

A set of clear notation for the style of diagram to the left of Figure 4.6 would be a valuable adaptation of the plait presentation format. If we were then able to implement this in a programming language we could make drastic savings on the amount of work done by a program, and hence reduce the time that it takes to complete a calculation. One possibility is to use Morse link presentations (similar notation can be seen in [59]); an instance of the information of this presentation being used for computing purposes can be seen in [34]. While there is always going to be some work involved in first obtaining a diagrammatic presentation for a knot as a plait or Morse link presentation, and in writing out how the information of such a presentation may be encoded, it will always be more simple to do so than to calculate a polynomial invariant of the diagram by hand.

4.7.5 Implementation in a compiled language

We have considered algorithms for both the Kauffman polynomial and the Homfly polynomial, and implemented both of them in Maple. While this is useful to show that the algorithm can be implemented in a computing language, Maple is not without its flaws for running the kinds of operations used in the approach that I have shown.

While the code that has been written is designed to work for a plait of width k, Maple's own capabilities make it unlikely that it could cope with an example beyond $k = 4$ for an implementation that considers operations on the whole set of k-sequences. An implementation of the algorithm in a compiled programming language like C++ could offer a lot.

Firstly, a better method for organising the storage of the set of k-sequences could be found, so that locating a k-sequence in memory might be an easier task than it currently is. More importantly, we could improve the management and storage of the coefficients that are passed from one k-sequence to another through the various operations that are performed. One reason why the calculation slows down (in the Maple implementations) is that it is a difficult process to store all of the coefficients, as well as organise the way that they are moved around in memory. This leads to the program slowing down for larger values of c, a situation which could be improved by implementing the algorithm in a compiled language.

4.8 Examples

The calculations in this section were performed on a computer with an AMD Duron 1.59GHz processor with 480MB of RAM, and using Maple 11 running on the University of Liverpool Managed Windows Service.

4.8.1 Alternating 3-plait family

We calculate the Homfly polynomials of a family of alternating links based around the presentation

$$\sigma_2{\sigma_3}^{-1}\sigma_4{\sigma_5}^{-1}({\sigma_1}^{-1}\sigma_2{\sigma_3}^{-1}\sigma_4{\sigma_5}^{-1})^{2n}{\sigma_1}^{-1}\sigma_2{\sigma_3}^{-1}\sigma_4$$

for $n \in \mathbb{N}$. See Table 4.1 for results of the calculations. We list the number of crossings in the presentation, the time taken by the program `h_plait` to calculate the Homfly polynomial and the bound on the braid index as given by Theorem 1.8.

These calculations and the calculations for the next example are important because of the bound on the braid index that we obtain (from Theorem 1.8). Previous polynomial time algorithms for the calculation of the Homfly polynomial have been based around a braid presentation of the knot. The braid index of the larger examples we calculate here are substantially greater than previous programs could handle.

Other programs exist that are based on general diagrams of knots, but these are limited in terms of the number of crossings that a diagram can have. Again, examples in the family of links that we have generated and calculated invariants for have substantially more crossings than previous programs could deal with.

4.8.2 Alternating 4-plait family

We calculate the Homfly polynomials of a family of alternating links based around the presentation

$$\sigma_2{}^{-1}\sigma_3\sigma_4{}^{-1}\sigma_5\sigma_6{}^{-1}\sigma_7(\sigma_1\sigma_2{}^{-1}\sigma_3\sigma_4{}^{-1}\sigma_5\sigma_6{}^{-1}\sigma_7)^{2m}\sigma_1\sigma_2{}^{-1}\sigma_3\sigma_4{}^{-1}\sigma_5\sigma_6{}^{-1}$$

for $n \in \mathbb{N}$. See Table 4.2 for results of the calculations. We list the number of crossings in the presentation, the time taken by the program `h_plait` to calculate the Homfly polynomial and the bound on the braid index.

n	c	Calculation time	MFW
0	8	0.240	3
1	18	0.161	6
2	28	0.501	8
3	38	1.141	11
4	48	2.453	14
5	58	4.076	16
6	68	8.413	19
7	78	15.752	22
8	88	15.312	24
9	98	22.372	27
10	108	31.345	30
11	118	39.697	32
12	128	52.905	35
13	138	80.776	38
14	148	104.471	40
15	158	158.418	43

Table 4.1: Calculation times and braid index bounds for alternating 3-plaits

m	c	Calculation time	MFW
0	12	1.272	4
1	26	3.405	7
2	40	11.467	11
3	54	30.113	15
4	68	66.616	18
5	82	105.022	21
6	96	172.437	25
7	110	277.690	29
8	124	502.713	32
9	138	539.627	35
10	152	780.252	39

Table 4.2: Calculation times and braid index bounds for alternating 4-plaits

Chapter 5

Genus 2 Mutation

5.1 Introduction

The work of this chapter appeared in a slightly different form in the paper "Invariants of genus 2 mutants" [44], and was inspired by a talk that I attended given by Alexander Shumakovitch, one of the authors of [15].

Genus 2 mutation of knots was introduced by Ruberman in a general 3-manifold [53]. Cooper and Lickorish gave an account of an equivalent construction for knots in S^3 using genus 2 handlebodies [13]; it is this construction that we use here.

Genus 2 mutant knots allow us to compare knot invariants; it can be shown that they share a certain collection of invariants, and thus any invariant on which some mutant pair differs must be completely independent of the shared collection. This procedure can be refined by restricting further the class of genus 2 mutants under consideration, so as to increase the shared collection, and then looking for invariants which differ on some restricted mutants.

A survey of some of the known results about shared invariants for genus 2 mutants is given in [15]. The authors also give an example of a pair of genus 2 mutants with 75 crossings with different Homfly polynomials. These are smaller examples than the known satellites of the Conway and Kinoshita-Teresaka knots [42].

The authors conjectured that their pair of knots did not share Kauffman polynomials, but calculations for knots of this complexity are out of range of current programs. In the absence of a calculation for their own knots they asked for examples of genus 2 mutants which do not share the Kauffman polynomial.

In this chapter we describe a pair of 55-crossing genus 2 mutant knots with different Homfly polynomials, and show without performing a direct calculation that they have different Kauffman polynomials. We show other interesting results for these examples regarding their Vassiliev invariants and quantum $sl(3)$ invariants. We note also a distinction between general genus 2 mutants and those arising as satellites of Conway mutant knots. Our 55-crossing pair of genus 2 mutants differ on a degree 7 Vassiliev invariant, while the work of [11] showed that satellites of Conway mutants share all Vassiliev invariants of degree ≤ 8. This was more recently extended by Jun Murakami [47], who showed that satellites of Conway mutants share all Vassiliev invariants up to degree 10.

We summarise the other examples of [44], giving some details of their Homfly and Kauffman polynomials, as well as their Vassiliev invariants. Finally we refer to a recent example of Stoimenow and Tanaka [57].

5.2 Genus 2 mutation

In Chapter 1 we defined mutation of knots and links in the standard sense. We now give a construction for genus 2 mutation, due to Ruberman [53].

Definition

Take a framed oriented curve P in the standard genus 2 handlebody W (P is framed as we use the framed Homfly relations).

Embed W in $\mathbb{R}^3$ by $h : W \to \mathbb{R}^3$, to get a curve $h(P) \subset \mathbb{R}^3$.

The π-rotation $\tau : W \to W$, illustrated in Figure 5.1, has 6 fixed points on ∂W, where it restricts to the hyperelliptic involution with quotient S^2. This lies in the centre of the mapping class group of ∂W and is unique up to conjugation by a homeomorphism isotopic to the identity.

Apply τ to P to get another curve $\tau(P) \subset W$. The curves $h(P)$ and $h(\tau(P))$ are called **genus 2 mutants**.

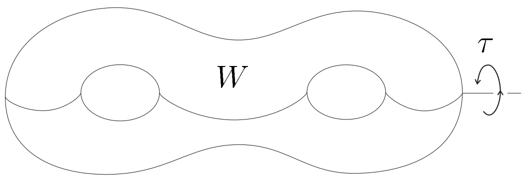


Figure 5.1: The rotation τ

Theorem 5.1 ([44]) *Satellites of genus 2 mutants are themselves genus 2 mutants.*

Theorem 5.2 ([44]) *Genus 2 mutants have the same Jones polynomial.*

Theorem 5.2 then shows, by Theorem 5.1, that satellites of genus 2 mutants cannot be distinguished by their Jones polynomials.

5.2.1 Genus 2 embeddings following a 2-tangle

In this section we establish the framework in which we consider many of the examples in this chapter. We will consider diagrams of a certain type (see Figure 5.7) in order to separate the curve P and the embedding for the knot, and use these to study genus 2 mutation.

We distinguish two types of oriented 2-tangle:

1. A *pure* tangle, where the arcs join the two bottom points to the corresponding top points on the same side.

2. A *transposing* tangle, where the arcs join the two bottom points to the top points on opposite sides.

We now show how to use a framed oriented 2-tangle F to define an embedding $h : W \to \mathbb{R}^3$ in such a way that we can readily compare the framed curves $h(P)$ and $h(\tau(P))$.

Let W be the thickening, $S \times I$, of a standard surface S, and define h by thickening a map from S to S_F.

To specify h we assume that F has a framing, that is each arc has a specified ribbon neighbourhood. Define a surface S_F in $\mathbb{R}^3$ consisting of a square plus two ribbons following the framing of F. Figure 5.2 shows an example with the tangle from the Conway/Kinoshita-Teresaka knots.

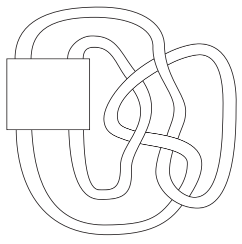


Figure 5.2: The surface following a framed tangle

Our choice of S, and hence the description of h, depends on the nature of the tangle F. When F is a pure tangle the surface S_F is a disc with 2 holes. Take S to be the square with two ribbons in Figure 5.3 and map S to S_F by taking the square to the square, and the two ribbons to the ribbons around the arcs of F.

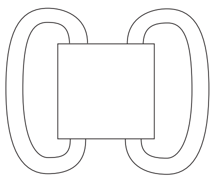


Figure 5.3: The disc with 2 holes

When F is a transposing tangle the surface S_F is a torus with one hole. Take S to be the square with two ribbons in Figure 5.4 and again map S to S_F by mapping the square to the square, and the ribbons around the arcs of F.

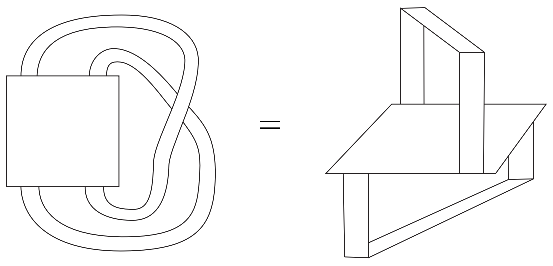


Figure 5.4: The torus with one hole

We say that h has been constructed by *following* the tangle F. An embedded handlebody in $\mathbb{R}^3$ always arises by following some tangle F, although the choice of F is not unique.

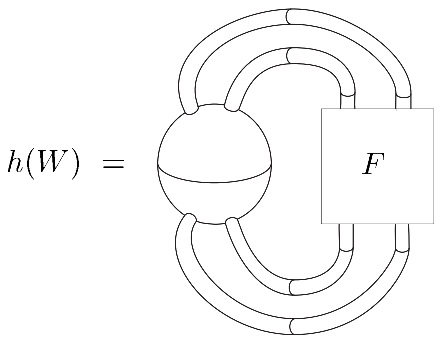


Figure 5.5: The handlebody following a tangle F

We can get a good view of the pair of mutants constructed from a curve $P \subset W$ by following a tangle F. The map $\tau : W \to W$ is a thickened map from S to S, which maps the square and each ribbon to itself.

In the case of pure tangles, τ is π-rotation about the horizontal x-axis, which we write as τ_1 when restricted to the square. For transposing tangles, τ is π-rotation about the z-axis orthogonal to the plane of the square, and we write τ_2 for this rotation restricted to the square. These rotations are indicated in Figure 5.6.

Draw P as a diagram on the surface S, so that its framing is the blackboard framing from S. We can assume that P runs through each ribbon of S in a number of parallel curves, possibly with different orientations.

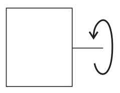
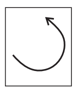
$\tau_1 \;=\;$  , $\tau_2 \;=\;$  .

Figure 5.6: Rotations of the square

Suppose that there are m_1 curves in one ribbon and m_2 in the second, numbered from the attachment to the top edge of the square. The rest of the curve P determines a framed m-tangle T in the square, with $m = m_1 + m_2$.

For a pure tangle F, the knot $h(P)$ has a diagram as shown in Figure 5.7, where $F^{(m_1,m_2)}$ is the (m_1, m_2) parallel of the framed tangle F with appropriate orientations, and the tangle T lies in the square. For a transposing tangle F, the knot $h(P)$ has a diagram as shown in Figure 5.8, where $F^{(m_1,m_2)}$ is the (m_1, m_2) parallel of the framed tangle F with appropriate orientations, and the tangle T lies in the square.

Proposition 5.3 *When h follows a pure tangle, the genus 2 mutant knot $h(\tau(P))$, has $\tau_1(T)$ in place of T, with all orientations in $F^{(m_1,m_2)}$ reversed. When h follows a transposing tangle, the genus 2 mutant knot $h(\tau(P))$ has $\tau_2(T)$ in place of T.*

Proof
For a pure tangle, τ_1 is the appropriate rotation applied to T. Reversing orientations does not effect the Homfly polynomial, and ensures that orientations are aligned correctly.

For a transposing tangle, τ_2 is the appropriate rotation applied to T. ∎

5.2.2 Conway mutants

In section 1.6 we introduced the idea of mutation of knots, as first introduced by Conway [12]. We give a slightly different definition here, formally defining the rotations of the tangles.

Definition

For an oriented tangle T write $\tau_1(T)$ and $\tau_2(T)$ for the π-rotations of T

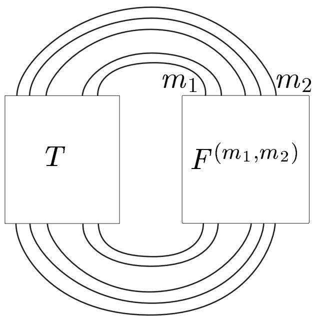


Figure 5.7: The diagram for a knot following a pure tangle F

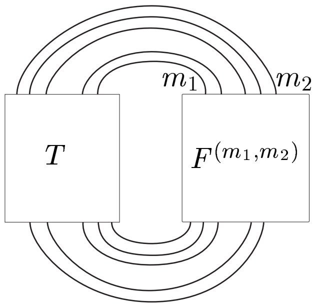


Figure 5.8: The diagram for a knot following a transposing tangle F

about the x-axis and z-axis respectively, as used above. Then $\tau_3(T) = \tau_1\tau_2(T)$ is the π-rotation of T about the y-axis, so that

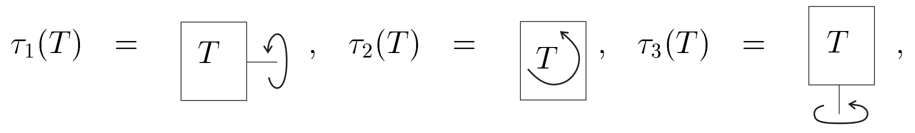


Figure 5.9: Rotations for Conway mutation

A knot K can be decomposed into two oriented 2-tangles F and G as in Figure 5.10. Any knot K' formed by replacing the tangle F with the tangle $F' = \tau_i(F), i = 1, 2, 3$, reversing its string orientations if necessary is called a **mutant** of K, or a **Conway mutant** of K.

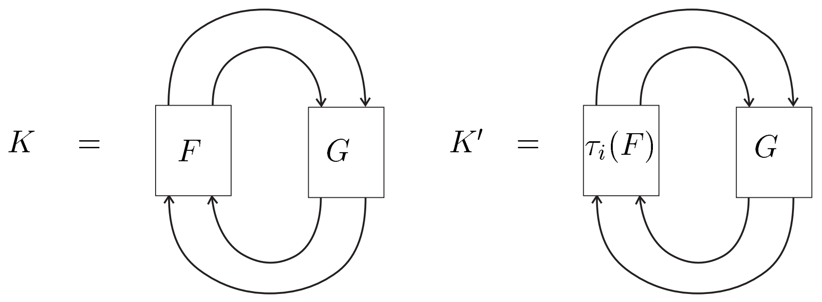
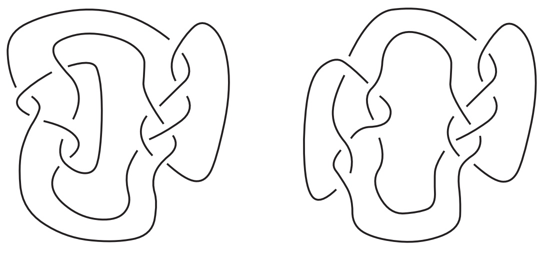


Figure 5.10: A knot with mutants

The two 11-crossing knots in Figure 5.11 are the best-known example of a pair of mutant knots; these knots were presented with different diagrams in Figure 1.14.

5.2.3 Conway mutants as genus 2 mutants

Any knot K made up of two 2-tangles F and G as in Figure 5.10 lies in two genus 2 handlebodies, one following F and the other following G. Each of these handlebodies defines a genus 2 mutant of K. We call them K_F and K_G respectively.



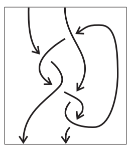
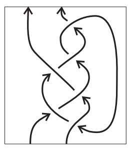
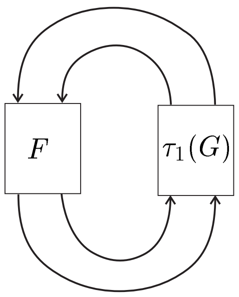
$F =$  , $G =$  , $F' = \tau_3(F)$.

Figure 5.11: The Conway and Kinoshita-Teresaka mutant pair, and their constituent tangles

Since K is a knot, specifically a link of one component, one of the tangles of F and G is pure and the other is transposing. Suppose that F is pure. Then K_F and K_G have diagrams as shown in Figure 5.12.

$K_F =$ 

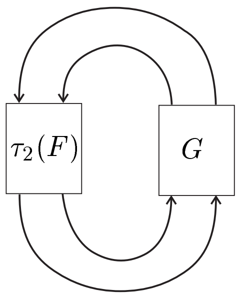
$K_G =$ 

Figure 5.12: Genus 2 mutants of K

We can repeat the construction on these knots. The knot K_F lies in the handlebody following $\tau_1(G)$. Since $\tau_1(G)$ is transposing we get a genus 2 mutant $K_{F\tau_1(G)}$. The same knot $K_{G\tau_2(F)} = K_{F\tau_1(G)}$ arises as a genus 2 mutant of K_G from the handlebody following $\tau_2(F)$, shown in Figure 5.13.

Proposition 5.4 *Up to a choice of string orientation the three knots K_F, K_G and $K_{F\tau_1(G)}$ are the three Conway mutants of K given by replacing F with $\tau_1(F), \tau_2(F)$ or $\tau_3(F)$ respectively.*

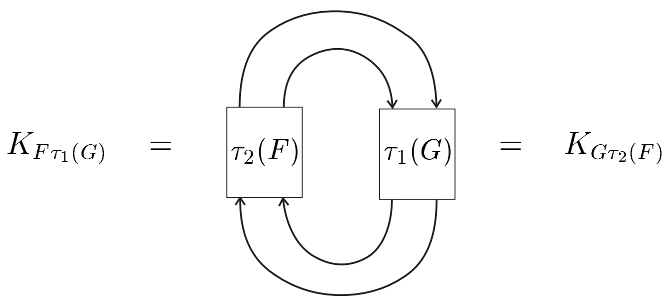


Figure 5.13: A further genus 2 mutant, completing the Conway mutants of K

Proof
By comparing the diagrams with those resulting from the rotations of F it is clear that they are the Conway mutants. ∎

It follows that satellites of Conway mutants, with this orientation convention, are related by genus 2 mutation.

5.3 Homfly polynomials of genus 2 mutants

We use the framed version of the Homfly polynomial based on the skein relations given in subsection 1.5.2 with the substitution $z = s - s^{-1}$.

The Homfly polynomial of a link in $\mathbb{R}^3$ is unchanged if the orientations of all its components are reversed (Lemma 1.6). The Homfly skein of the annulus $\mathcal{C}$ is unchanged when the annulus is rotated by π, reversing its core orientation, and at the same time all string orientations are reversed [19].

Thus in order to compare the Homfly polynomials of two genus 2 mutants $h(P)$ and $h(\tau(P))$, or indeed any satellite of them, it is enough to consider $h(\tau(P))$ with orientation reversed.

Given a framed oriented curve P in W we may regard W as the thickened surface S which is the disc with 2 holes in Figure 5.3, and compare P with $\tau(P)$ after reversing the orientation of $\tau(P)$. If we can present P as an $(m_1 + m_2)$-tangle in the square with m_1 and m_2 curves following the two ribbons then we can write P in the skein of the twice-punctured disc S as a linear combination of simpler curves, each presented by a tangle with at most this number of curves in the ribbons.

Even if our curve P has originally been drawn in a picture following a transposing tangle, with m_1 and m_2 curves around the ribbons there, it can be redrawn as a curve following a pure tangle with the same numbers m_1 and m_2.

If $m_1 = m_2 = 1$ then the genus 2 mutants are Conway mutants, and by Theorem 1.9 their Homfly polynomials agree.

In the case $m_1, m_2 \leq 2$ the curve P reduces in the skein of S to a combination of curves in the skein of S which are unchanged by the rotation τ with reversal of string orientation. This is essentially the result of Lickorish and Lipson [30]. There are a couple of cases depending on the relative orientation of the curves in the two ribbons. This argument covers the case of any 2-string satellite of a pair of Conway mutants, as these can be presented as genus 2 mutants with $m_1 = m_2 = 2$.

The existence of 3-string satellite knots around the Conway and Kinoshita-Teresaka mutant pair with different Homfly polynomials [42] (following earlier calculations by Morton and Traczyk) shows that there are genus 2 mutants with $m_1 = m_2 = 3$, constructed by following the constituent tangle G in Figure 5.10, which have different Homfly polynomials.

Take, for example, the tangle T to be the 3-parallel $F^{(3,3)}$ of the tangle F in Figure 5.10 composed with the braid $\sigma_1\sigma_2$ and follow the tangle G to give a knot with 101 crossings. This is in fact a satellite of the Conway knot, whose genus 2 mutant has $\tau_2(T)$ in place of T.

5.4 Kauffman polynomials of genus 2 mutants

The pair of 75 crossing genus 2 mutants given in [15] were shown to have different Homfly polynomials, and the coefficients were given explicitly in the paper. The authors of [15] were unable to calculate the Kauffman polynomials for their 75 crossing examples, constructed following the pure 7-crossing tangle DG shown in Figure 5.14.

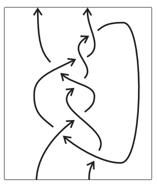
$DG =$ 

Figure 5.14: The 7-crossing tangle DG

As noted previously, it is a computationally difficult task to calculate knot polynomials; the Kauffman polynomial is more difficult to calculate in general than the Homfly polynomial.

However, given the Homfly polynomials of two knots, there is an indirect method that we can potentially use to show that their Kauffman polynomials differ, and in particular we can use this method in the case of genus 2 mutation.

Denote the constant part of the Homfly polynomial of a knot by $P_0(v)$ (i.e., the coefficient in v of z^0). Similarly denote the constant part of the Kauffman polynomial of a knot by $D_0(v)$. The following result will be very useful for the examples we give in the rest of this chapter.

Lemma 5.5 ([28]) *For any knot,* $P_0(v) = D_0(v)$.

If $P_0(v)$ differs for a pair of knots then $D_0(v)$ differs also and hence the Kauffman polynomials differ. Hence if $P_0(v)$ differs for a pair of genus 2 mutants then $D_0(v)$ differs also and hence the Kauffman polynomials of the genus 2 mutants differ. This argument could not be used for the pair of knots in [15], as the Homfly polynomials of their knots had the same $P_0(v)$ term.

The remainder of the work of this chapter is given to examples of pairs of genus 2 mutants with differing Kauffman polynomials; in all of these cases we have shown indirectly that the Kauffman polynomials of the pairs differ because their $P_0(v)$ terms differ.

We also give some details of the Vassiliev invariants of our examples, and some information on their quantum $sl(3)$ invariants.

5.5 Main Result

Inspired by the combinatorial interpretations of the $v = s^3$ substitution in Homfly leading to the Kuperberg skein of the twice-punctured disc [43], we have found a pair of examples following DG with $m_1 = 3, m_2 = 2$ and orientations $++-$ and $+-$. The curve P is shown in Figure 5.15 as a diagram in the disc with two holes, S, along with the resulting 5-tangle T.

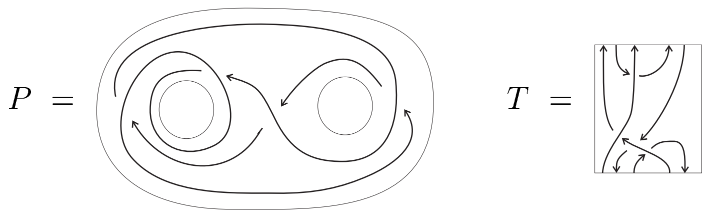


Figure 5.15: The curve P in the standard handlebody, and related tangle T

We construct two 55-crossing genus 2 mutants from P by following the tangle DG, to give the knot S_{55}, shown in Figure 5.16. Its mutant partner S'_{55} is given by a rotation of the tangle T.

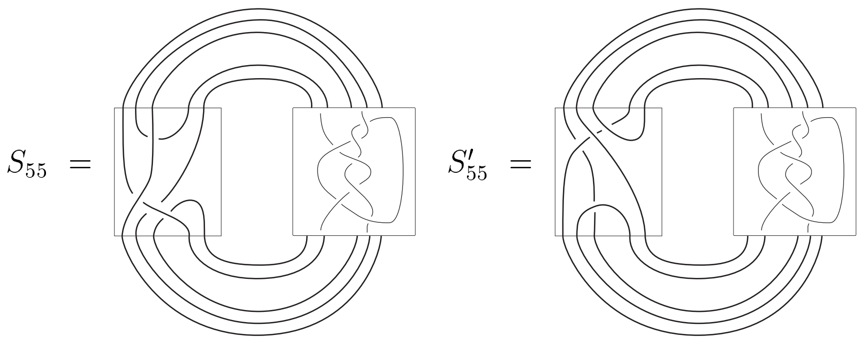


Figure 5.16: Two 55-crossing genus 2 mutants with different Homfly and Kauffman polynomials

Theorem 5.6 *The genus 2 mutant knots S_{55} and S'_{55} have different Homfly and Kauffman polynomials.*

Proof
The coefficients for the Homfly polynomials of S_{55} and S'_{55} are shown in Tables 5.1 and 5.2. They were calculated using the program of Imafuji and Ochiai [20], since the knots are not readily expressed as closed braids.

S_{55}	v^{-4}	v^{-2}	1	v^2	v^4	v^6	v^8	v^{10}	v^{12}
1	-36	122	-143	67	-23	32	-23	5	
z^2	-276	986	-1199	550	-148	223	-172	34	3
z^4	-757	3003	-3884	1811	-345	567	-478	75	20
z^6	-1048	4688	-6531	3158	-400	718	-690	76	45
z^8	-827	4243	-6360	3217	-253	499	-585	39	34
z^{10}	-388	2355	-3774	1985	-87	192	-302	10	10
z^{12}	-107	814	-1386	746	-15	38	-92	1	1
z^{14}	-16	171	-308	166	-1	3	-15		
z^{16}	-1	20	-38	20			1		
z^{18}		1	-2	1					

Table 5.1: Coefficients of the Homfly polynomial of S_{55}

Immediately we can see that they have different Homfly polynomials. The first row of coefficients in each table gives the value $P_0(v)$, and so Lemma 5.5 shows that S_{55} and S'_{55} have different Kauffman polynomials. ∎

Corollary 5.7 *The Homfly polynomials of S_{55} and S'_{55} still differ after the substitution $v = s^3$, and their Vassiliev invariants differ at degree 7.*

Proof
We can look at $sl(3)$ invariant information as a Laurent polynomial in s by making the substitutions $z = s - s^{-1}$, $v = s^3$. The difference is:

$$s^{-24}\left(s^4 - s^2 + 1\right)\left(s^4 + s^3 + s^2 + s + 1\right)\left(s^4 - s^3 + s^2 - s + 1\right)\left(s^8 + 1\right)$$
$$\left(s^6 + s^5 + s^4 + s^3 + s^2 + s + 1\right)\left(s^6 - s^5 + s^4 - s^3 + s^2 - s + 1\right)$$
$$\left(s^2 + s + 1\right)^2\left(s^2 - s + 1\right)^2\left(s^4 + 1\right)^2\left(s^2 + 1\right)^3(s - 1)^8(s + 1)^8$$

S'_{55}	v^{-4}	v^{-2}	1	v^2	v^4	v^6	v^8	v^{10}	v^{12}
1	-38	-135	-178	-116	-58	-39	-16		1
z^2	257	924	1171	662	288	209	60	-34	-16
z^4	-687	-2591	-3205	-1587	-562	-448	-72	142	54
z^6	964	3913	4779	2080	566	509	24	-226	-73
z^8	-782	-3530	-4260	-1623	-319	-334	10	172	43
z^{10}	377	1991	2356	766	100	126	-7	-67	-11
z^{12}	-106	-709	-814	-213	-16	-25	1	13	1
z^{14}	16	155	171	32	1	2		-1	
z^{16}	-1	-19	-20	-2					
z^{18}		1	1						

Table 5.2: Coefficients of the Homfly polynomial of S'_{55}

The factor $(s-1)^8$ shows that they differ in a Vassiliev invariant of degree 8 invariant arising from $sl(3)$. However, we can obtain Vassiliev invariants for S_{55} and S'_{55} directly as the coefficients of powers of h in the power series given by substituting $z = e^{\frac{h}{2}} - e^{-\frac{h}{2}}$, $v = e^{\frac{Nh}{2}}$. The lowest term in the difference of the power series for S_{55} and S'_{55} is

$$3N(N-1)(N-2)(N-3)(N+3)(N+2)(N+1)h^7,$$

so these differ in a Vassiliev invariant of degree at most 7. ∎

The 75 crossing examples from [15] have Vassiliev invariants that differ at degree 11; we calculated the difference at that degree to be

$$N\,(N-1)\,(N-2)\,(N+2)\,(N+1)\,\big(13\,N^2+51\big)\,h^{11}$$

using the same substitutions and method as previously.

Their examples use a 6-tangle with $m_1 = m_2 = 3$, where the orientations of the three strands around one of the ribbons are $++-$ while around the other they are $+++$. As with the example of our 55 crossing knots, the Homfly polynomials of their 75 crossing knots remain different when $v = s^3$, however this was not shown in [15]. The difference, as a Laurent polynomial in s, is:

$$\begin{aligned}&s^{-28}\left(s^4-s^2+1\right)\left(s^4+s^3+s^2+s+1\right)\left(s^4-s^3+s^2-s+1\right)\\&\quad\left(s^8+1\right)\left(s^6+s^5+s^4+s^3+s^2+s+1\right)\left(s^6-s^5+s^4-s^3+s^2-s+1\right)\\&\quad\left(s^2-s+1\right)^2\left(s^2+s+1\right)^2\left(s^4+1\right)^2\left(s^2+1\right)^3(s-1)^{11}(s+1)^{11}\end{aligned}$$

In the preparation of [44] we had originally tried to make use of the difference from the $v = s^3$ substitution of the 75-crossing examples to show that the Kauffman polynomials were different. We planned to argue through the comparison of the Homfly polynomials of a certain 2-string satellite at $v = s^4$, without actually calculating this Homfly polynomial, which would be well out of range. Our aim was to make use of a comparison in [37] between this evaluation of the satellite invariant and a different evaluation of the Kauffman polynomial of the original knots, knowing something of the evaluations of the satellite invariant at $v = s^3$.

Unfortunately the difference in the invariants at $v = s^3$ contains a factor $(s^6 + s^5 + s^4 + s^3 + s^2 + s + 1)$ which means that the agreement of the evaluations of the satellite at $v = s^4$ can not be excluded. This has also proved to be the case in any other examples that we have found where the evaluations at $v = s^3$ are different, so there may be some underlying reason for this.

5.6 Other Results

5.6.1 A 72 crossing example

Theorem 5.8 *The genus 2 mutant pair of knots constructed by following the tangle DG, with $m_1 = m_2 = 3$, using the 6-string positive permutation braid $\beta = \sigma_1\sigma_2\sigma_1\sigma_3\sigma_2\sigma_4\sigma_3\sigma_5\sigma_4$ or its reverse $\tau_1(\beta)$ as the tangle T, have different Kauffman polynomials.*

Proof

The two knots are presented as closed 9-braids with 72 crossings, so it is quite easy to calculate their Homfly polynomials using the Morton-Short program based on the Hecke algebras [45]. When these are compared they can be seen to differ in their constant term $P_0(v)$. By Lemma 5.5 the constant terms of their Kauffman polynomials differ, and hence their Kauffman polynomials differ. ∎

In the 72 crossing examples the string orientations around each ribbon are all in the same sense $+ + +$, and as a result the knots have the same Homfly invariant after the substitution $v = s^3$. This is a general consequence of the analysis of the Kuperberg skein of the surface S in [43] for the case $m_1 = m_2 = 3$ in which all the orientations around the ribbons are $+$.

The Vassiliev invariants for our 72 crossing examples differ at degree 7:

$$3N(N-1)(N-2)(N-3)(N+3)(N+2)(N+1)h^7.$$

Consequently satellites of Conway mutants share more Vassiliev invariants than general genus 2 mutants, since they have all Vassiliev invariants of degree ≤ 10 in common, using the result from [42] that Vassiliev invariants of degree $\leq k$ of a satellite $K * Q$ are Vassiliev invariants of K of the same degree, and Jun Murakami's result [47] about Vassiliev invariants of Conway mutants.

5.6.2 A 56 crossing example

The pair of 56-crossing genus 2 mutants following the transposing Conway tangle G with 6 crossings, using the 6-braid $\sigma_2\sigma_3$ and its rotation $\tau_2(\sigma_1\sigma_2) = \sigma_3\sigma_4$ with $m_1 = m_2 = 3$, are shown in Figure 5.17. These are closed 9-braids related to Conway and Kinoshita-Teresaka satellites.

Like our 72-crossing examples in Theorem 5.8 it can be shown indirectly that this pair have different Kauffman polynomials, by calculating their Homfly polynomials and then taking advantage of Lemma 5.5. They also differ in a degree 7 Vassiliev invariant, but share the same value when $v = s^3$.

5.6.3 Further examples

Various examples using the Conway tangle G as in Figure 5.17 with values $m_1 = 2$ and $m_2 = 3$ were tried in order to generate pairs of genus 2 mutants.

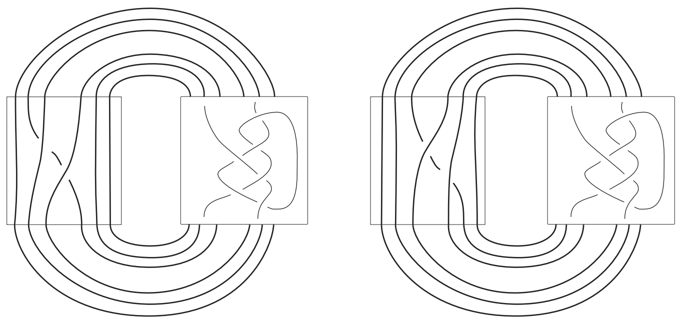


Figure 5.17: Two closed 9-braid genus 2 mutants with different Homfly polynomial

Some of these examples had fewer than 50 crossings, but none of the examples that were tried had differing Homfly polynomials. It remains to be seen if examples of genus 2 mutants with differing Homfly and Kauffman polynomials can be found that have fewer than 55 crossings.

We have been unable to compute Kauffman directly for any of the examples that we have shown, and have always relied on Lemma 5.5 and a differing $P_0(v)$ value in the calculated Homfly polynomials.

The starting point for this investigation was the example of [15], and our initial approach was to attempt to indirectly calculate the difference of the Kauffman polynomials of the mutant pair. Using the theory of manipulating stacked tangles in the Kauffman skein (as in Chapter 3) we were able to show a non-zero difference at the level of tangles by expressing $T - \tau_2(T)$ as a linear combination of stacked 6-tangles. While we were able to use this to express the difference of the original pair of knots as a sum of simpler diagrams, some of which had fewer than twenty crossings, it was still not possible to directly calculate the values of the larger diagrams in this linear combination.

Thus while we have been able to show that the Kauffman polynomials of genus 2 mutants can differ, we were unable to answer the first question posed in [15], and it is unknown whether the Kauffman polynomials of the 75-crossing examples differ.

5.7 A recent result

A recent paper of Stoimenow and Tanaka gives a pair of 56-crossing knots related by genus 2 mutation with differing Homfly and Kauffman polynomials, although the authors do not refer to them as genus 2 mutants [57]. The examples are Whitehead doubles of the 14-crossing genus 2 mutants 14_{41721} and 14_{42125}. The 14-crossing knots have presentations in a genus 2 handlebody with $m_1 = 2$ and $m_2 = 1$, and so have identical Homfly and Kauffman polynomials.

The authors of [15] also use the same pair of 14-crossing knots to show a result in Khovanov homology, and they are referenced in [44]. The knots follow the pure tangle AB in Figure 5.18 and use the curve P, shown in Figure 5.19 as a diagram in the disc with two holes along with the resulting 3-tangle T.

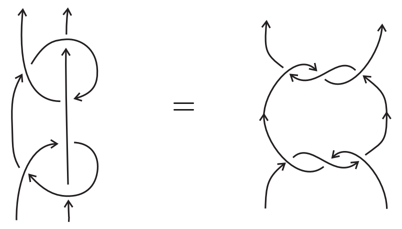


Figure 5.18: The tangle AB used in [15]

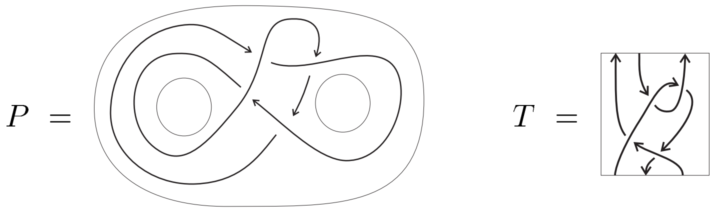


Figure 5.19: A curve P, and related tangle T

By Theorem 5.1 any of their satellites will be related by genus 2 mutation also, and so the pair of knots that Stoimenow and Tanaka calculated knot polynomials for give another example of genus 2 mutants with differing Homfly and Kauffman polynomials.

The $P_0(v)$ term is identical for the two 14-crossing knots so we cannot deduce indirectly that they have different Kauffman polynomials: it can be shown from a skein theoretic argument that if $P_0(v)$ coincides for two knots then it will coincide for any satellites of those knots.

The authors of [57] were able to calculate the Kauffman polynomials of the Whitehead doubles of 14_{41721} and 14_{42125} almost directly. They showed that the Kauffman polynomials of the 2-cables of the 14-crossing knots were different, and then by a skein theoretic argument they were able to show that the Kauffman polynomials of the Whitehead doubles of the knots would differ.

As with all of our examples save for S_{55} and S'_{55}, there is no difference in the Homfly polynomials of these examples with substitution $v = s^3$, although they do differ with the substitution $v = s^4$. The Vassiliev invariants differ at degree 11 as follows:

$$4N^3\,(N-1)\,(N-2)\,(N-3)\,(N+3)\,(N+2)\,(N+1)\,h^{11}.$$

5.8 Discussion

There are several areas of interest arising from the work of this chapter, and from the area of polynomial invariants of genus 2 mutation. The examples of Theorem 5.6 provide 55-crossing genus 2 mutants with differing Homfly and Kauffman polynomials. These appear to be the smallest examples in the literature in terms of crossing number.

In searching for smaller examples we know that such pairs of genus 2 mutants must have a certain degree of complexity. As stated earlier, genus 2 mutants with $m_1, m_2 \leq 2$ are guaranteed to have identical Homfly and Kauffman polynomials. When $m_1 = 3$, $m_2 = 2$ we have the first instance that we can hope to see differing polynomials; this naturally leads to a reasonably high lower bound on the number of crossings that a knot must have for it to be one of a pair of genus 2 mutants with differing Homfly and Kauffman polynomials. In the preparation of [44] examples of genus 2 mutants with as few as 40 crossings were examined, but they did not differ on their Homfly polynomials.

As the $P_0(v)$ values of these smaller examples were identical an assessment as to whether or not their Kauffman polynomials differed could not be made. An interesting question that I believe is open is whether genus 2 mutants with differing Homfly polynomials are guaranteed to have differing Kauffman polynomials. Similarly, if a pair of genus 2 mutants have identical Homfly polynomials does that mean that they will have identical Kauffman polynomials?

Our 55-crossing knots and the 75-crossing knots of [15] both have differing Homfly polynomials after the substitution $v = s^3$, but our other examples and the example of [57] do not; both our example and the example of [15] have the feature that they follow the tangle DG. Further investigation into the pure and transposing tangles that one uses in constructing these examples might lead to an answer.

Finally, we note that our three examples differ at degree 7 for Vassiliev invariants. This is in contrast both to the examples of [15] and [57], which differed at degree 11, and to the general theory for Conway mutants, where it is known that Vassiliev invariants must agree up to degree 10 [47]. Firstly, what is different about our examples compared to the examples of [15] and [57] that allows an earlier difference in Vassiliev invariants? Secondly, how do Vassiliev invariants behave in general for genus 2 mutants? The result of [47] guarantees that genus 2 mutants that result from satellites of Conway mutants must have Vassiliev invariants agreeing up to degree 10, but we know very little about the Vassiliev invariants of genus 2 mutants in general.

Chapter 6

Kauffman Polynomials of Pretzel Links

6.1 Introduction

Pretzel links are an interesting class of links for study. They have a regular structure, and it is easy to give notation for describing them.

Reidemeister first considered them [51], and pretzels have been used many times to show certain properties of knots or links. Trotter used them to show that non-invertible knots exist [58]; Landvoy gave an easily implemented algorithm for calculating the Jones polynomial [27], and more recently Morton used the construction to show some interesting results in mutation [40].

In this chapter, we take advantage of the regular structure of pretzels to construct an algorithm for calculating the Kauffman polynomial of pretzel links. Theorem 6.2 starts by showing that we can express the Kauffman polynomial of a pretzel diagram as a linear combination of the Kauffman polynomials of much simpler diagrams; later in the chapter we use the term “elementary pretzel” to denote these diagrams and show that by placing some restrictions on these diagrams we obtain a good algorithm for calculating the Kauffman polynomial.

This algorithm is easily implemented in Maple, and in principle it is more efficient than an algorithm that works on a naive approach on the number of crossings in a diagram. The algorithm operates by calculating certain coefficients from recurrence relations; while we are able to obtain generating functions from these recurrence relations we note at the end of the chapter that the generating functions pose problems when implemented in Maple.

6.2 Pretzel Links

Definition

A **pretzel** link is given by a sequence of half twists connected in a certain way, as in the example of Figure 6.1. General pretzels can be represented by a k-tuple $(p_1, p_2, \ldots, p_k)$, $k \geq 3$, $p_i \in \mathbb{Z}$, $1 \leq i \leq k$. $|p_i|$ is the number of half twists, and the sign of p_i denotes whether the $|p_i|$ half twists are right-handed or left-handed (L_+ or L_- respectively). Figure 6.2 gives this more general form of $(p_1, p_2, \ldots, p_k)$.

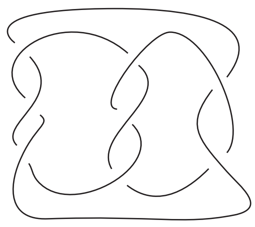


Figure 6.1: The pretzel $(3, 3, -2)$

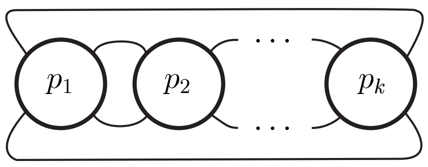


Figure 6.2: The pretzel $(p_1, p_2, \ldots, p_k)$

We take $k > 2$, as $k = 1$ would give a diagram which is a twisted unknot, and $k = 2$ would give a diagram which is a closed 2-braid.

Theorem 6.1 *If k is odd and all of the p_i are odd then a knot is produced. If k is even and all of the p_i are odd then a two component link is produced. Else the number of even p_i gives the number of components for k both even or odd.*

Proof
The first two cases can be realised by considering how one travels around the diagram starting from a point. The third case can be shown simply by observing that two even p_i in a k-tuple have a link component between them; we can draw a circle between each of the even p_i to represent a link component, and if the number of even p_i is m it is not difficult to see that there will be m circles and hence m components. ∎

Hence, a k-tuple denotes a knot if and only if k is odd and all of the p_i are odd, or if there is exactly one even p_i. In all other cases the k-tuple gives a link.

In general, permutation of the p_i coding for a knot results in knots related by mutation, and hence these will have identical Kauffman polynomials. Permuting the p_i of a 3-pretzel always results in an isotopic link. This can be observed simply from the structure of 3-pretzels.

6.3 Twists in the Kauffman skein

The regular structure of pretzels suggests that there might be some shortcut that we can take over the general approach that the skein relations give us for calculating the Kauffman polynomial. The approach of this chapter is to express $|p_i|$ half-twists as a linear combination of single half-twists and the two smoothings, with coefficients from the Kauffman skein. We can construct recursive formulae for the coefficients of these linear combinations, and these give a method of easily expressing n half-twists as a linear combination of three elements.

Using these formulae on the sequences of half-twists within a pretzel structure we get a linear combination of much simpler diagrams: we trade one diagram with a large number of crossings for many diagrams with far fewer crossings.

In the following discussion we borrow the language of braid groups (with only a few small abuses) to express the crossings in the half twists; we take σ to be a single right-handed crossing, σ^{-1} to be a single left-handed crossing, e to be the identity represented by L_0 and h to be the L_∞ smoothing. We consider the following actions to be taking place in the Kauffman skein algebra of $(2, 2)$-tangles.

Theorem 6.2 *The Kauffman polynomial of a pretzel link $(p_1, p_2, \ldots, p_k)$ can be expressed as a linear combination of the Kauffman polynomials of at most 3^k diagrams, each with at most k crossings, with coefficients from the Kauffman skein.*

Proof

We first show that we can express n half twists as a linear combination of single half twists and smoothings. We write the main Kauffman skein relation as

$$\sigma - \sigma^{-1} = z\,(e - h),$$

and by the Kauffman skein relation for framing we see

$$\begin{array}{rcccl} \sigma h & = & h\sigma & = & v\sigma \\ \sigma^{-1}h & = & h\sigma^{-1} & = & v^{-1}h \end{array}$$

Now consider the following rearrangement:

$$\begin{aligned} \sigma - \sigma^{-1} &= z\,(e - h) \\ \sigma &= \sigma^{-1} + ze - zh \\ \sigma^2 &= \sigma^{-1}\sigma + ze\sigma - zh\sigma \\ \sigma^2 &= e + z\sigma - vzh \\ &= z\sigma + e - vzh. \end{aligned}$$

Thus we have a relation for expressing σ^2 in terms of σ, e and h with coefficients in v and z from the Kauffman skein.

Lemma 6.3 *n right-handed crossings, written σ^n, can be expressed as a linear combination of a single right-handed crossing σ and smoothings e and h in the following way*

$$\sigma^n = f_1(n)\sigma + f_1(n-1)e + f_3(n)h$$

where f_1 and f_3 are recurrence relations defined by

$$\begin{aligned} f_1(n) &= zf_1(n-1) + f_1(n-2) \qquad & f_1(0) = 0,\ f_1(1) = 1 \\ f_3(n) &= v(f_3(n-1) - zf_1(n-1)) \qquad & f_3(1) = 0 \end{aligned}$$

Proof
From our result for σ^2 there is no doubt that we can construct a recursive method for calculating an expression for σ^n in terms of σ, e and h, so we need only show what form this relation takes. Initially define

$$\sigma^n = f_1(n)\sigma + f_2(n)e + f_3(n)h$$

where f_1, f_2 and f_3 are recurrence relations for polynomials in v and z.

Take the expression for the case of σ^{n-1} and multiply both sides of the expression by σ. We then use the result for σ^2 in order to evaluate the expression further.

$$\begin{aligned} \sigma^{n-1} &= f_1(n-1)\sigma + f_2(n-1)e + f_3(n-1)h \\ \sigma^n &= f_1(n-1)\sigma^2 + f_2(n-1)e\sigma + f_3(n-1)h\sigma \\ &= f_1(n-1)(z\sigma + e - vzh) + f_2(n-1)\sigma + vf_3(n-1)h \\ &= (zf_1(n-1) + f_2(n-1))\sigma + f_1(n-1)e \\ &\quad + v(f_3(n-1) - f_1(n-1))h. \end{aligned}$$

We compare the two expressions for σ^n and evaluate the recurrence relations as

$$\begin{aligned} f_1(n) &= zf_1(n-1) + f_2(n-1) \\ f_2(n) &= f_1(n-1) \\ f_3(n) &= v(f_3(n-1) - zf_1(n-1)). \end{aligned}$$

The relation $f_2(n)$ is in terms of $f_1(n)$, and hence the recurrence relation for $f_1(n)$ is more helpfully written as

$$f_1(n) = zf_1(n-1) + f_1(n-2).$$

Consequently, our expression for σ^n can be written as

$$\sigma^n = f_1(n)\sigma + f_1(n-1)e + f_3(n)h,$$

and from results already known we can state the initial conditions for these recurrence relations:

$$\begin{array}{rcl} f_1(n) & = & zf_1(n-1) + f_1(n-2) \qquad f_1(0) = 0,\ f_1(1) = 1 \\ f_3(n) & = & v(f_3(n-1) - zf_1(n-1)) \qquad\quad f_3(1) = 0 \end{array}$$

∎

From the Kauffman skein relation we obtain

$$\sigma^{-2} = -z\sigma^{-1} + e + v^{-1}zh.$$

As before we will be able to find an expression for σ^{-n} in terms of the expression for $\sigma^{-(n-1)}$, and so on, back to the expression we have for σ^{-2}. As with the case for σ^n we work by comparing the general case for σ^{-n} with the expression for $\sigma^{-(n-1)}$ multiplied by σ^{-1}. This leads us to the following result which we state without proof.

Lemma 6.4 *n left-handed crossings, written σ^{-n}, can be expressed as a linear combination of a single left-handed crossing σ^{-1} and smoothings e and h in the following way*

$$\sigma^{-n} = g_1(n)\sigma^{-1} + g_1(n-1)e + g_3(n)h$$

where g_1 and g_3 are recurrence relations defined by

$$\begin{array}{rcl} g_1(n) & = & g_1(n-2) - zg_1(n-1) \qquad g_1(0) = 0,\ g_1(1) = 1 \\ g_3(n) & = & v^{-1}(zg_1(n-1) + g_3(n-1)) \qquad\quad g_3(1) = 0 \end{array}$$

With Lemma 6.3 and Lemma 6.4 we are in a position to prove Theorem 6.2.

Consider the diagram of a pretzel given by $(p_1, \ldots, p_k)$. By Lemma 6.3 and Lemma 6.4, for each p_i, we can express the $|p_i|$ half-twists as a linear combination of three different diagrams. These are a single crossing (right-handed or lefthanded), and the two possible smoothings from the Kauffman skein.

Applying these results to each p_i gives a linear combination of at most three terms. Taken over the k twists this then gives a total of at most 3^k different diagrams in the sum. One diagram in the sum will have a single crossing in each of the places, resulting in a diagram with k crossings. The other $3^k - 1$ diagrams will have fewer crossings. ∎

The upper bound on the number of diagrams, 3^k, is sharp if and only if $|p_i| > 1$ for all $1 \leq i \leq k$.

It is worth noting that unless all of the p_i are of the same sign, the diagram with k crossings mentioned in the proof of Theorem 6.2 can be simplified further using Type II Reidemeister moves.

The recurrence relations are simple to mechanise in a computing language. It is relatively straight forward to realise some code that will calculate the coefficients for the terms in the expressions of σ^n and σ^{-n}. In general recurrence relations can be quite intensive procedures to run, but in Maple we can add the code “`option remember`” which generates a table of values as the procedure runs. We gain the illusion of speed at the expense of storing values in memory.

We give some code for calculating these coefficients later in the chapter.

6.4 Cubic Relation

There is a cubic relation that we can show for the right-handed crossing σ. We have to rearrange to remove h from the expressions that we build up (using the rearrangement $h = e - \frac{1}{z}\sigma + \frac{1}{z}\sigma^{-1}$).

$$\begin{aligned}\sigma &= \sigma^{-1} + ze - zh\\ \sigma^2 &= z\sigma + e - vzh\end{aligned}$$

We replace h with the terms in e, σ and σ^{-1}, and then take e (as an identity element) to have value 1. Then

$$\begin{aligned}
\sigma^2 &= z\sigma + e - vzh \\
\sigma^2 &= z\sigma + 1 - vz(1 - \frac{1}{z}\sigma + \frac{1}{z}\sigma^{-1}) \\
\sigma^2 &= z\sigma + 1 - vz + v\sigma - v\sigma^{-1}
\end{aligned}$$

and by multiplying through by σ and collecting terms

$$\begin{aligned}
\sigma^3 &= z\sigma^2 + \sigma - vz\sigma + v\sigma^2 - v \\
\sigma^3 &= (z+v)\sigma^2 + (1 - vz)\sigma - v.
\end{aligned}$$

We take the specialisation $z = s - s^{-1}$, and then rearrange to give a cubic equation in σ with coefficients in v and z:

$$\sigma^3 - (s - s^{-1} + v)\sigma^2 - (1 - v(s - s^{-1}))\sigma + v = 0.$$

This factorises to give

$$-\frac{1}{s}(\sigma s + 1)(-s + \sigma)(-\sigma + v) = 0$$

which has roots $\sigma = -s^{-1}$, $\sigma = s$, $\sigma = v$.

The roots of this equation give us a way of defining generating functions for the coefficients. However, in the method that we will outline this will not be helpful due to the elimination of the term in h. Later in the chapter we will consider generating functions obtained from the recurrence relations.

6.5 Elementary Pretzels

We now turn our attention to the simpler diagrams that result from the application of Theorem 6.2.

Definition

An **elementary pretzel** is given by a sequence $r = [r_1, r_2, \ldots, r_k]$, where the r_i are elements from the set $\{+1, -1, 0, \infty\}$ and represent respectively a righthanded crossing, a left-handed crossing, the smoothing L_0 and the smoothing L_∞. The sequence r defines a diagram in a similar way to the k-tuples that give pretzel diagrams. The r_i are thought of diagrammatically as being in the same location as the p_i in the definition of pretzels.

Consider the diagram of the elementary pretzel $[+1, +1, -1, \infty, 0]$ as in Figure 6.3. The value of this diagram in the Kauffman skein is v^{-1}, but in general we could have a more difficult knotted structure.

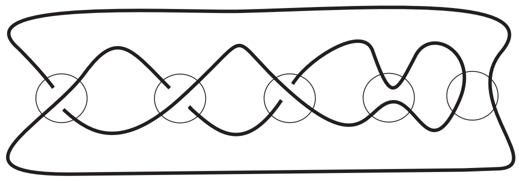


Figure 6.3: Diagram for elementary pretzel $[+1, +1, -1, \infty, 0]$

Consider the rotation of Figure 6.3 through 90 degrees. If we had diagrams that did not contain the smoothing L_0 then by rotating an elementary pretzel through 90 degrees we could see easily the number of crossings that the diagram actually contained. Due to the simple structure that such diagrams have, the Kauffman polynomial of this diagram could be realised as a simple sum of twisted or disjoint unknots with coefficients provided by the recurrence relations we have already evaluated.

Proposition 6.5 *n half twists, whether right-handed or left-handed, can be represented as a linear combination of right-handed and left-handed crossings, and the smoothing L_∞ represented by the element h. The coefficients of these three terms can be obtained from the recurrence relations established in Lemmas 6.3 and 6.4.*

Proof

The main Kauffman skein relations have four terms, and so we can always express any linear combination of these four elements in terms of at most three of them. Thus in expressing σ^n and σ^{-n} in terms of single crossings and smoothings we can eliminate terms in e.

Then

$$\begin{aligned}
\sigma^n &= f_1(n)\sigma + f_1(n-1)e + f_3(n)h \\
&= f_1(n)\sigma + f_1(n-1)(z^{-1}\sigma - z^{-1}\sigma^{-1} + h) + f_3(n)h \\
&= (f_1(n) + z^{-1}f_1(n-1))\sigma - z^{-1}f_1(n-1)\sigma^{-1} \\
&\quad + (f_3(n) + f_1(n-1))h
\end{aligned}$$

and

$$\begin{aligned}
\sigma^{-n} &= g_1(n)\sigma^{-1} + g_1(n-1)e + g_3(n)h \\
&= g_1(n)\sigma^{-1} + g_1(n-1)(z^{-1}\sigma - z^{-1}\sigma^{-1} + h) + g_3(n)h \\
&= z^{-1}g_1(n-1)\sigma + (g_1(n) - z^{-1}g_1(n-1))\sigma^{-1} \\
&\quad + (g_3(n) + g_1(n-1))h,
\end{aligned}$$

taking the same values for the recurrence relations as defined previously in Lemmas 6.3 and 6.4. ∎

Corollary 6.6 *The Kauffman polynomial of a pretzel $p = (p_1, \ldots, p_k)$ can be expressed as a linear combination of the Kauffman polynomials of 3^k elementary pretzels of the form $[r_1, \ldots, r_k]$ where the r_i are elements of the set $\{+1, -1, \infty\}$.*

Proof
Applying Proposition 6.5 to the proof of Theorem 6.2 shows this result. ∎

Definition

Let r_+ be the number of right-handed crossings in an elementary pretzel r, and r_- be the number of left-handed crossings.

Definition

For an elementary pretzel, r, without the smoothing L_0 we obtain a diagram r_N by rotating r through 90 degrees and viewing it as in Figure 6.4. This allows us to see the number of crossings and the handedness of these crossings, which we can obtain from r as $N = r_- - r_+$.

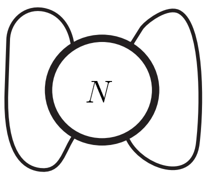


Figure 6.4: Diagram r_N

Lemma 6.7 *The Kauffman polynomial of a diagram r_N is*

$$D(r_N) = \begin{cases} \delta & N = 0 \\ f_1(N)v + f_1(N-1)\delta + f_3(N) & N > 0 \\ g_1(N)v^{-1} + g_1(N-1)\delta + g_3(N) & N < 0 \end{cases}$$

with relations f_1, f_3, g_1 and g_3 defined as previously.

Proof

The diagram r_0 is a pair of disjoint unknots, and so has value δ as defined in Section 1.5.3. The Kauffman polynomial of r_N for $N \neq 0$ is easy to calculate using the recurrence relations of Lemmas 6.3 and 6.4. Applying these formulae to a diagram r_N will result in a linear combination of at most three diagrams, these being the unknot, a twisted unknot, and two disjoint unknots. ∎

We restate Theorem 6.2 as Theorem 6.8.

Theorem 6.8 *The Kauffman polynomial of a pretzel $p = (p_1, p_2, \ldots, p_k)$ can be expressed as a linear combination of the Kauffman polynomials of diagrams of the form r_N, where N varies between $-k$ and k.*

Proof

By Corollary 6.6 we express p as a linear combination of 3^k elementary pretzels in the Kauffman skein. Each of these elementary pretzels can be expressed as some diagram of the form r_N. The values for N are derived from the possible elementary pretzels of length k: we express the sum of 3^k elementary pretzels as a linear combination of the $2k+1$ possible diagrams of the form r_N where N varies between $-k$ and k. ∎

6.6 Algorithm

We combine the various results that we have shown in this chapter to give an algorithm for calculating the Kauffman polynomial of pretzel links.

For each p_i in a sequence for a pretzel $p = (p_1, \ldots, p_k)$ we calculate the coefficients from representing those $|p_i|$ half twists as a linear combination of the elements σ, σ^{-1} and h.

Effectively we are obtaining the information that we need to express the diagram given by the k-tuple p as a linear combination of 3^k elementary pretzels $r = [r_1, \ldots, r_k]$ and which have coefficients from the Kauffman skein given by certain products of the coefficients obtained by evaluating the p_i.

Expressing the p_i as a linear combination of the elements σ, σ^{-1} and h means that the 3^k elementary pretzels of Corollary 6.6 will be given by all of the possible elementary pretzels of length k where the terms r_i are from elements in the set $\{+1, -1, \infty\}$.

The Kauffman polynomial of each of these 3^k diagrams is now easily calculable if we consider them to be in the format of Figure 6.4. By calculating the Kauffman polynomials of these r_N we complete the calculation of the Kauffman polynomial of the pretzel link $p = (p_1, \ldots, p_k)$.

This is a simple algorithm to consider on paper, but the coefficients will be much too unwieldy to calculate invariants of any non-trivial examples by hand. The algorithm is readily implemented in a programming language. In the next section we give an example of a series of Maple procedures that lead to an implementation for calculating the Kauffman polynomial of a pretzel.

6.7 Implementation

The most straight forward way to implement this algorithm, I believe, is to start with the recurrence relations that we defined earlier, and then build up the program piece by piece. We use these relations in other procedures, which do more and more complicated things but continue to look relatively simple. Eventually we are able to give the main routine which performs the algorithm, calling in the relative sub-procedures as necessary.

The benefit of this approach is that the main routine is relatively clear, and is not cluttered with overly complicated expressions and lines of code.

6.7.1 Recurrence Relations

We begin by giving the procedures for the four recurrence relations (from Lemmas 6.3 and 6.4). These are the foundation of the algorithm, and so are of great importance in the implementation. The line of code "`option remember`" in each routine improves the speed of the procedures by creating a table of previously calculated values. We gain the illusion of speed in calculation by increasing memory use to store these values.

```
f1 := proc(n::nonnegint)
  option remember:
  if n=1 then return 1: end if:
  if n=0 then return 0: end if:
  return expand(z * f1(n-1) + f1(n-2));
end proc:

f3 := proc(n::nonnegint)
  option remember:
  if n=1 then return 0: end if:
  return expand(v * (f3(n-1) - z * f1(n-1)));
end proc:

g1 := proc(n::nonnegint)
  option remember:
  if n=1 then return 1: end if:
  if n=0 then return 0: end if:
  return expand(g1(n-2) - z * g1(n-1));
end proc:
```

```
g3 := proc(n::nonnegint)
  if n=1 then return 0: end if:
  return expand( (1/v) * (z * g1(n-1) + g3(n-1)));
end proc:
```

With these procedures we have the foundations of an implementation of the algorithm.

6.7.2 Building Up Procedures

We create procedures which return triples of coefficients for σ^n and σ^{-n}, when they are expressed as linear combinations of σ, σ^{-1} and h.

```
SIGMAn := proc(n::posint)
local output:
  output := [0,0,0]:
  #output[1] is the coeff of {sigma}
  output[1] := expand( (1/z) * f1(n+1) ):
  #output[2] is the coeff of {sigma}^(-1)
  output[2] := expand( -(1/z) * f1(n-1) ):
  #output[3] is the coeff of h
  output[3] := expand( f3(n) + f1(n-1) ):
  output;
end proc:

SIGMA_n := proc(n::posint)
local output:
  output := [0,0,0]:
  #output[1] is the coeff of {sigma}
  output[1] := expand( (1/z) * g1(n-1) ):
  #output[2] is the coeff of {sigma}^(-1)
  output[2] := expand( - (1/z) * g1(n+1) ):
  #output[3] is the coeff of h
  output[3] := expand( g3(n) + g1(n-1) ):
  output;
end proc:
```

Note that in both SIGMAn and SIGMA_n we could set the values directly as we define the triple output; however, by writing the code in the manner that I have given it is clear how we are arriving at these coefficients.

The entries for output[1] in SIGMAn and output[2] in SIGMA_n have been slightly simplified by considering the recurrence relations.

The p_i in a k-tuple for a pretzel can be positive or negative. Rather than use SIGMAn and SIGMA_n directly in the main routine it is simpler if we have a smaller routine that will call the appropriate procedure to deliver the output. One way that we can implement this is as follows.

```
Kcoeff := proc(n::integer)
local out:
  if n = 0 then
    out := [ 1/z, -1/z, 1 ]:
  elif n > 0 then
    out := SIGMAn(n):
  elif n < 0 then
    out := SIGMA_n(-n):
  end if:
out;
end proc:
```

As we will use Kcoeff in the calculation of the Kauffman polynomial of diagrams of the form r_N we include the possibility of an input of 0.

One final subroutine that we require is something that gives the value of N for a reduced diagram r in the format r_N.

Recall that $N = r_- - r_+$. In this implementation we denote right-handed crossings with $+1$, lefthanded crossings by -1 and the smoothing L_∞ by 0, as it does not contribute to the sum of crossings. Hence N is the sum of the entries in r multiplied by -1, and we can implement this function with the following routine.

```
r2N  := proc()
local t,i:
t := 0:
for i from 1 to nargs do t := t + args[i] end do:
-t;
end proc:
```

6.7.3 The Main Routine

With the procedures that we have built up, we are now in a position to implement the complete algorithm.

I have tried to give the implementation in as simple a manner as possible, and give a short outline after the listing of the program.

```
with(combinat, permute):
##permute required to generate the desired
##possible elementary pretzels of length k
pretzel := proc()
local A,L,M1,M,N,i,j,k,C,store,total:
k := nargs:
L := [seq(1,i=1..k), seq(-1,i=1..k), seq(0,i=1..k)]:
M1 := permute(L,k):   M := Array(1..nops(M1)):
for i from 1 to nops(M1) do M[i] := M1[i] end do:
##M represents the set of elementary pretzels
##of length k where each r_i is a crossing or h
M1 := 'M1':   C := [args]:
for i from 1 to k do C[i] := Kcoeff(C[i]) end do:
total := 0:
for i from 1 to ArrayNumElems(M) do
  ##for each elementary pretzel
  store := 0:   N := r2N(op(M[i])):   A := Kcoeff(N):
  store := expand((1/v)*A[1]+v*A[2]+A[3]):
  ##in the loop the initial assignment for store is a
  ##calculation of the Kauffman polynomial for some
  ##diagram r_N
  for j from 1 to k do
    if M[i][j] = 1 then
      store := expand(store*C[j][1]):
    elif M[i][j] = -1 then
      store := expand(store*C[j][2]):
    elif M[i][j] = 0 then
      store := expand(store*C[j][3]):
    end if:
  end do:
  ##the previous loop calculates the contribution
  ##to the coefficient of each of the r_i, passed from
  ##the linear combination of the p_i
  total := expand(total+store):
end do:
collect(expand(total),z);
end proc:
```

The procedure works by first producing a list of all of the possible sequences $r = [r_1, \dots, r_k]$, where the r_i are elements of the set $\{+1, -1, \infty\}$. These sequences are the elementary pretzels we will consider. Then the coefficients of expressing each of the p_i as a linear combination of σ, σ^{-1} and h are calculated. We sum over the set of the r we have established; we multiply by the appropriate coefficients resulting from the calculations of the expressions of the p_i and calculate the Kauffman polynomials of the reduced diagrams r by considering them in the format r_N.

As we have developed the procedure `Kcoeff` it is simpler to use this to calculate the coefficients of the linear combination of twisted unknots that result from calculating the Kauffman polynomial of a diagram r_N, rather than use the function we defined previously in Lemma 6.7.

6.7.4 Remark

Permuting the p_i for a pretzel link does not change the Kauffman polynomial, as permuting the p_i is the same as performing mutations on the link. Thus we can consider performing calculations with the set of twists: the order is not important.

One way that we might improve our calculations is to reorder the sequence $(p_1, p_2, \ldots, p_k)$ so that we first consider the positive p_i ordered to be strictly non-decreasing, and then the negative p_i so that they are strictly non-increasing. In this manner we can build up a table of results (`option remember` in Maple) in an organised way to minimise the number of calculations performed.

6.8 Generating Functions

While the algorithm that we have developed certainly has its advantages over a naive approach to calculating a knot polynomial, the use of recurrence relations to calculate coefficients is inefficient. Their use in the implementation only gives the illusion of fast calculation, and without the "`option remember`" lines of code in each of the recurrence relations the implementation would take much longer to compute the Kauffman polynomial of even a relatively simple example.

Generating functions should allow for a much faster calculation time. We can derive these from the recurrence relations that we have already realised, but must use the specialisation of variables $z = s - s^{-1}$.

Theorem 6.9 *For $n \in \mathbb{N}$ we can obtain the following generating functions for coefficients from the recurrence relations of Lemmas 6.3 and 6.4*

$$\begin{aligned} f_1(n) &= \frac{s^n - (-s^{-1})^n}{s + s^{-1}} \\ f_3(n) &= \frac{v(s - s^{-1})}{(s^{-1} + v)(s - v)} v^n - \frac{v(s - s^{-1})}{s + s^{-1}} \left(\frac{s^n}{s - v} + \frac{(-s^{-1})^n}{s^{-1} + v} \right), \end{aligned}$$

and the roles of functions g_1 and g_3 are filled since

$$\begin{aligned} f_1(-n) &= f_1(n)_{|s \to s^{-1}} \\ f_3(-n) &= f_3(n)_{\big| {s \to s^{-1} \atop v \to v^{-1}}}. \end{aligned}$$

Proof

We derive these generating functions from the recurrence relations by using some relatively simple theory, and using the specialisation $z = s - s^{-1}$. We get the generating functions for f_1 and g_1 first, as these are involved in the expressions for g_1 and g_3 respectively. We then solve non-homogeneous recurrence relations to obtain the generating functions for f_3 and g_3. Initially, we obtain the following functions for the recurrence relations:

$$\begin{aligned} f_1(n) &= \frac{s^n - (-s^{-1})^n}{s + s^{-1}} \\ f_3(n) &= \frac{v(s - s^{-1})}{s + s^{-1}} \left(\frac{1}{s^{-1} + v}(v^n - (-s^{-1})^n) + \frac{1}{s - v}(v^n - s^n) \right) \\ g_1(n) &= \frac{(s^{-1})^n - (-s)^n}{s + s^{-1}} \\ g_3(n) &= \frac{v^{-1}(s - s^{-1})}{s + s^{-1}} \left(\frac{1}{s^{-1} - v^{-1}}((s^{-1})^n - (v^{-1})^n) + \frac{1}{s + v^{-1}}((-s)^n - (v^{-1})^n) \right) \end{aligned}$$

We perform some rearrangements and collect terms for f_3 and g_3 that make them simpler.

$$\begin{aligned} f_3(n) &= \frac{v(s - s^{-1})}{(s^{-1} + v)(s - v)} v^n - \frac{v(s - s^{-1})}{s + s^{-1}} \left(\frac{s^n}{s - v} + \frac{(-s^{-1})^n}{s^{-1} + v} \right) \\ g_3(n) &= \frac{v^{-1}(s^{-1} - s)}{(s + v^{-1})(s^{-1} - v^{-1})} (v^{-1})^n + \frac{v^{-1}(s^{-1} - s)}{s + s^{-1}} \left(\frac{(s^{-1})^n}{s^{-1} - v^{-1}} + \frac{(-s)^n}{s + v^{-1}} \right) \end{aligned}$$

Comparing f_1 and g_1, and f_3 and g_3, we can easily observe that we obtain g_1 and g_3 by making a substitution in the expressions for f_1 and f_3. Hence

$$\begin{aligned} g_1(n) &= f_1(n)_{|s \to s^{-1}} \\ g_3(n) &= f_3(n)_{\left| \substack{s \to s^{-1} \\ v \to v^{-1}} \right.} \end{aligned}$$

and thus we only need to use one set of functions and make substitutions to obtain the output of the others, since the recurrence relations g_1 and g_3 are calculating coefficients for left-handed twisting we state

$$\begin{aligned} f_1(-n) &= f_1(n)_{|s \to s^{-1}} \\ f_3(-n) &= f_3(n)_{\left| \substack{s \to s^{-1} \\ v \to v^{-1}} \right.}, \end{aligned}$$

as required. ∎

6.8.1 Remark

These substitutions also allow us to give a statement for the recurrence relations for the coefficients. Since $s \to s^{-1}$ and $z = s - s^{-1}$ we note that for polynomials in v and z

$$\begin{aligned} f_1(-n) &= f_1(n)_{|z \to -z} \\ f_3(-n) &= f_3(n)_{\left| \substack{z \to -z \\ v \to v^{-1}} \right.}. \end{aligned}$$

6.8.2 Implementation

The same approach is taken to the algorithm as before, the only difference being that we now have a different method for calculating coefficients. Rather than have four separate relations that we rely on, we have two functions. These calculate coefficients for the case that we have right-handed twists and we make a simple substitution by Theorem 6.9 in order to calculate coefficients for the case that we have left-handed twists ($p_i < 0$).

Thus the procedures for f_1 and f_3 are updated, and the routines `SIGMAn`, `SIGMA_n` and `Kcoeff` all have slight modifications. The main routine given previously is only altered to give terms in s and not z.

```
f1 := proc(n::nonnegint)
  (s^n - (-s^(-1))^n)/(s + s^(-1));
end proc:

f3 := proc(n::nonnegint)
  v*(s - s^(-1))/((s^(-1) + v)*(s - v))*v^n
  - v*(s - s^(-1))/(s + s^(-1))
     *((s^n/(s-v))+(((-s)^(-1))^n/(s^(-1)+v)));
end proc:

SIGMAn := proc(n::posint)
local output:
  output := [0,0,0]:
  #output[1] is the coeff of {sigma}
  output[1] := expand((1/(s-s^(-1)))*f1(n+1)):
  #output[2] is the coeff of {sigma}^(-1)
  output[2] := expand(-(1/(s-s^(-1)))*f1(n-1)):
  #output[3] is the coeff of h
  output[3] := expand(f3(n) + f1(n-1)):
  output;
end proc:

SIGMA_n := proc(n::posint)
local output:
  output := [0,0,0]:
  #output[1] is the coeff of {sigma}
  output[1] := expand((1/(s-s^(-1)))*subs(s=s^(-1),f1(n-1))):
  #output[2] is the coeff of {sigma}^(-1)
  output[2] := expand(subs(s=s^(-1),(1/(s-s^(-1)))*f1(n+1))):
  #output[3] is the coeff of h
  output[3] := expand(subs(s=s^(-1),v=v^(-1),f3(n) + f1(n-1))):
  output;
end proc:
```

```
Kcoeff := proc(n::integer)
local out:
  if n = 0 then
    out := [1/(s-s^(-1)), -1/(s-s^(-1)), 1]:
  elif n > 0 then
    out := SIGMAn(n):
  elif n < 0 then
    out := SIGMA_n(-n):
  end if:
out;
end proc:
```

The coefficients previously calculated by g_1 and g_3 are now calculated by making the substitution realised in Theorem 6.9 in to the expressions calculated by f_1 and f_3.

As noted previously we can use a substitution to reduce the number of recurrence relations that we use in an implementation of the algorithm. Were we to do this the only additional changes we would need to make would be in the routine `SIGMA_n`, in order to put the necessary substitutions in place.

6.8.3 Speed of calculation

In principle, using generating functions should give a quicker approach to calculating the invariant than by using an implementation that relies on recurrence relations. As noted previously, the recurrence relations that we have implemented only have the illusion of fast calculation because we create a table of values that calculations draw on in order to short circuit later calculations. Having to only perform one operation should then give generating functions an advantage over the recurrence relations in an implementation.

Based on calculations that I have performed, the opposite seems to be true: when comparing calculation times between two implementations, one based on recurrence relations and the other based on generating functions, we actually see the implementation based on recurrence relations greatly outperforming the implementation based on generating functions. This happens even when computing the Kauffman polynomial of simple pretzels with very few crossings.

I believe that the reason for this is that we are now calculating a polynomial in s and v, where $z = s - s^{-1}$. By doing so we are creating much larger polynomials that must be stored in memory, and this is slowing down the operation of Maple in what would otherwise be a simple enough calculation thanks to the theory that we have developed for calculating polynomial invariants for this family of knots.

6.8.4 Note

While I was writing up this chapter I became aware of a recently published paper on the Kauffman polynomials of pretzel links by Lu and Zhong [32]. Their method is different from mine, and does not approach the calculation through recurrence relations based on the twists in the pretzel links.

Chapter 7

The Skein of the Annulus

7.1 Introduction

In this chapter I present some preliminary calculations in the Kauffman skein of the annulus. While I was able to achieve some success in finding explicit values, I was unable to progress to a point where I could state a general result. We are able to make some reasonable conjectures on what might be true in a more general setting.

The work in this chapter follows work of [19] and [38] in investigating the skein of the annulus with two boundary points. In both of these papers the authors were considering the Homfly skein of the annulus and the skein of the annulus with two boundary points. In this chapter we see preliminary results that we have obtained through explicit manipulation and calculation of braid words with respect to the main Kauffman skein relations, and relations that we can derive from the interaction of elements in the annulus.

We look at linear combinations of closed braids on n strings in the skein of the annulus with an arc connecting points on the boundary. We show that certain linear combinations of braids in this setting can be expressed as linear combinations of identity braids on n strings and fewer than n strings.

7.2 Notation

7.2.1 The annulus

We consider elements in the annulus as in Figure 7.1. We take linear combinations of braid words X from $\mathbb{B}_n$ and close them. We take linear combinations with respect to the Kauffman skein relations, and we take these skein relations as defined in Section 1.5.3.

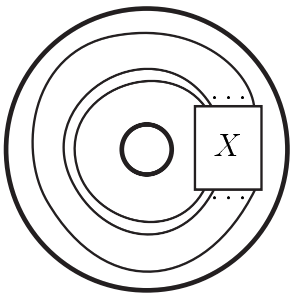


Figure 7.1: X, linear combination of words from $\mathbb{B}_n$

7.2.2 The annulus with two boundary points

Following the notation of [19] and [38] we give some initial constructions and definitions for the Kauffman skein of the annulus.

Denote by $\mathcal{K}$ the Kauffman skein of the annulus with two boundary points, one on each boundary component, as indicated in Figure 7.2.

The skein $\mathcal{K}$ becomes an algebra under the product induced by placing one annulus outside the other; for this, of course, we require that there is one curve connecting the two points on the boundary. The identity element in the skein, which we denote $a^0 \in \mathcal{K}$ to avoid confusion with the identity element of a braid, can be thought of as a single arc connecting the boundary points as in Figure 7.3.

Further elements are given by single arcs which wind around the central excluded point; the element a^1 is given by an arc that winds around the central

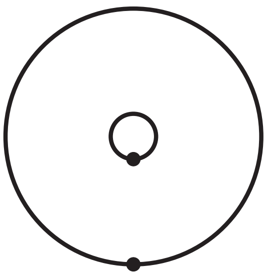


Figure 7.2: $\mathcal{K}$, the Kauffman skein of the annulus with two boundary points

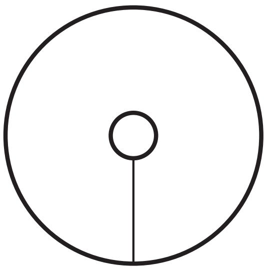


Figure 7.3: The element a^0

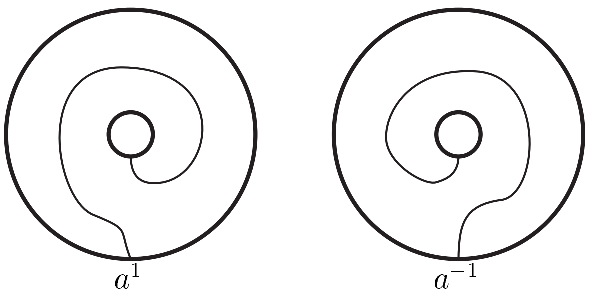


Figure 7.4: a^1 and a^{-1}

excluded point once in a counter-clockwise direction as we travel along it from the centre of the annulus to the outer boundary. This element can be seen in Figure 7.4 along with its inverse a^{-1}.

Powers of the element a^1, a^m for $m \in \mathbb{Z}$ are given by a single arc connecting the inner boundary point to the outer by winding in a counter-clockwise direction m times without crossing itself. We compose two elements by placing one annulus outside another, connecting arcs and boundary points; this action is commutative.

7.2.3 $l(X)$ and $r(X)$

The calculations that we wish to perform take place in the skein of the annulus with two boundary points. We consider two settings, and in both of these cases we have a linear combination, X, of words from $\mathbb{B}_n$ and an arc from the inner boundary to the outer boundary.

Define the settings $l(X)$ and $r(X)$ as in Figure 7.5.

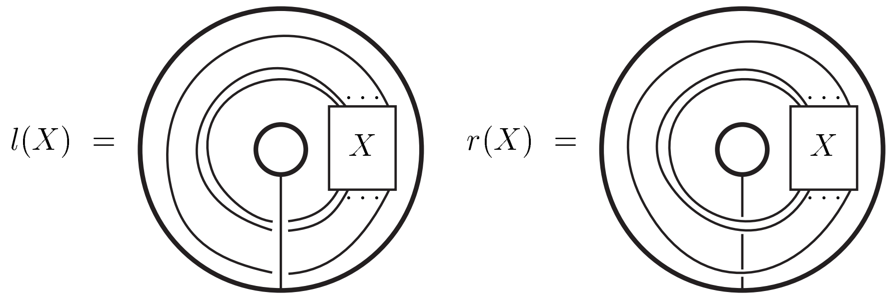


Figure 7.5: Settings $l(X)$ and $r(X)$

The notation introduced here mirrors some of the constructions in [38]. The theory of that paper was more developed in showing results for the Homfly skein of the annulus than the results for Kauffman in this chapter; however, I believe that the results in this chapter point the way to showing that similar results could be obtained for Kauffman.

We give a definition now that will make our later calculations easier to order.

Definition

For a linear combination, X, of braid words from $\mathbb{B}_n$, and for $0 \leq k \leq n$ we have the family of settings $r_k(X)$ in the annulus, where k gives the number of braid strings that the arc crosses under from the interior boundary point to the exterior; the arc passes under k consecutive braid strings, and then passes over the remaining $n - k$ strings.

We see how the arc connects the boundary points for $r_k(X)$ in Figure 7.6.

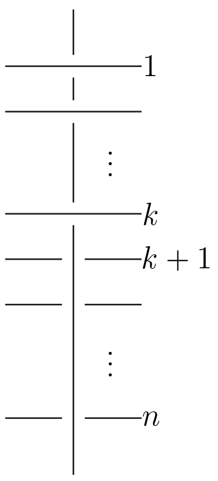


Figure 7.6: The arc connecting boundary points in the setting $r_k(X)$

Thus $r_0(X) = l(x)$ and $r_n(X) = r(X)$.

The object of this work is to consider expressing the elements $l(X) - r(X)$, for some X, as a sum of elements a^m with $m \in \mathbb{Z}$, $-n \leq m \leq n$. We are going to examine several cases of a specific family of examples for each n, which will give rise to some conjectures on the behaviour in general.

7.2.4 $P_n(X)$ and $N_n(X)$

Two other settings that we will need to consider in the annulus are $P_n(X)$ and $N_n(X)$, as seen in Figure 7.7.

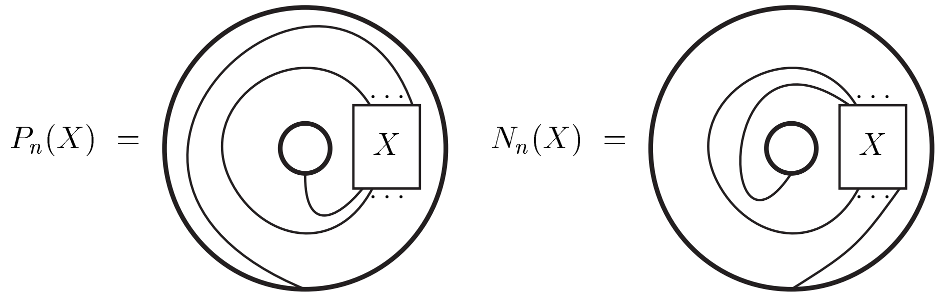


Figure 7.7: Settings $P_n(X)$ and $N_n(X)$

These settings are closer to the format of the elements that we wish to express our starting linear combinations as, i.e., they more closely resemble elements of the form a^m, $m \in \mathbb{Z}$.

7.2.5 Y_n

Definition

We define Y_n to be the linear combination of n words from the braid group $\mathbb{B}_n$ expressed as

$$\sigma_{n-1} \ldots \sigma_2\sigma_1 + \sigma_{n-1}^{-1} \ldots \sigma_2\sigma_1 + \ldots + \sigma_{n-1}^{-1} \ldots \sigma_2^{-1}\sigma_1 + \sigma_{n-1}^{-1} \ldots \sigma_2^{-1}\sigma_1^{-1}.$$

Thus Y_1 is simply the identity (and only) 1-braid, while Y_2 is $\sigma_1 + \sigma_1^{-1}$, and $Y_3 = \sigma_2\sigma_1 + \sigma_2^{-1}\sigma_1 + \sigma_2^{-1}\sigma_1^{-1}$. These are the examples that we shall consider explicitly in this chapter; we will make some reference to calculations for Y_4 and for Y_n in general, but we will not consider explicit calculations for $n > 3$. These examples follow on from work of Morton [38].

7.3 Calculations for Y_1 and Y_2

Calculations for Y_1 are almost trivial. Consider $l(Y_1)$ and $r(Y_1)$ as shown in Figure 7.8.

As Y_1 is the identity 1-braid the only difference between the two diagrams is from the crossing resulting from the arc connecting the boundary points.

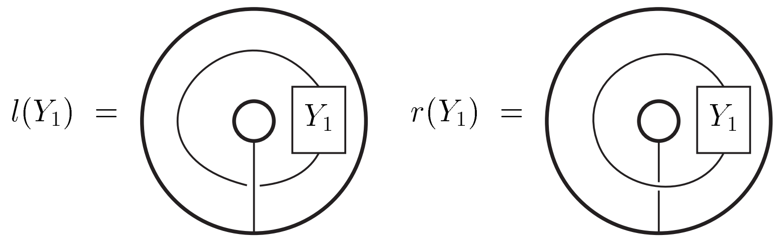


Figure 7.8: $l(Y_1)$ and $r(Y_1)$

Lemma 7.1 $l(Y_1) - r(Y_1) = z(a^1 - a^{-1})$.

Proof
This follows by applying the Kauffman skein relation to $l(Y_1) - r(Y_1)$. ∎

A valid intermediate point in the calculation for Y_1 would be to write the expression as $z(P_1(Y_1) - N_1(Y_1))$ after applying the skein relation, and then noting that this is the same as $z(a^1 - a^{-1})$.

The calculations for Y_2 are not completely trivial, and they require us to consider the diagrams that result from expressing $l(Y_2) - r(Y_2)$ as a series of diagrams.

Lemma 7.2 $l(Y_2) - r(Y_2) = z(z^2 + 4)(a^2 - a^{-2})$.

Proof
To begin with note

$$\begin{aligned} l(Y_2) - r(Y_2) &= r_0(Y_2) - r_2(Y_2) \\ &= (r_0(Y_2) - r_1(Y_2)) + (r_1(Y_2) - r_2(Y_2)). \end{aligned}$$

We consider $r_1(Y_2)$ as in Figure 7.9.

The diagram of $r_1(Y_2)$ differs from both $r_0(Y_2)$ and $r_2(Y_2)$ in exactly one place each, and we use the main Kauffman skein relation on each of the expressions $r_0(Y_2) - r_1(Y_2)$ and $r_1(Y_2) - r_2(Y_2)$. By considering the resulting diagrams we see the following,

$$r_0(Y_2) - r_1(Y_2) = z(P_2(\sigma_1 Y_2) - N_2(Y_2 \sigma_1^{-1}))$$

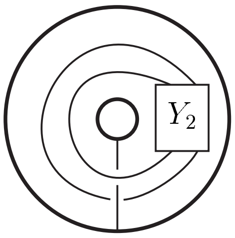


Figure 7.9: $r_1(Y_2)$

and

$$r_1(Y_2) - r_2(Y_2) = z(P_2(Y_2\sigma_1^{-1}) - N_2(\sigma_1 Y_2)).$$

Then we develop our previous expression as

$$\begin{aligned} l(Y_2) - r(Y_2) &= r_0(Y_2) - r_2(Y_2) \\ &= (r_0(Y_2) - r_1(Y_2)) + (r_1(Y_2) - r_2(Y_2)) \\ &= z(P_2(\sigma_1 Y_2) - N_2(Y_2\sigma_1^{-1})) + z(P_2(Y_2\sigma_1^{-1}) - N_2(\sigma_1 Y_2)) \\ &= z(P_2(\sigma_1 Y_2 + Y_2\sigma_1^{-1}) - N_2(\sigma_1 Y_2 + Y_2\sigma_1^{-1})). \end{aligned}$$

Now

$$\begin{aligned} \sigma_1 Y_2 + Y_2\sigma_1^{-1} &= \sigma_1(\sigma_1 + \sigma_1^{-1}) + (\sigma_1 + \sigma_1^{-1})\sigma_1^{-1} \\ &= \sigma_1^2 + e + e + \sigma_1^{-2} \\ &= \sigma_1^2 + \sigma_1^{-2} + 2e. \end{aligned}$$

In Chapter 6 we noted $\sigma^2 = z\sigma - vzh + e$ and $\sigma^{-2} = -z\sigma^{-1} + v^1 zh + e$ and we can adapt those results in this context to give

$$\begin{aligned} \sigma_1^2 + \sigma_1^{-2} + 2e &= z\sigma_1 - vzh_1 + e - z\sigma_1^{-1} + v^1 zh_1 + e + 2e \\ &= z(\sigma_1 - \sigma_1^{-1}) + z(v^{-1} - v)h_1 + 4e \\ &= z^2(e - h_1) + z(v^{-1} - v)h_1 + 4e \\ &= (z^2 + 4)e + z^2(\delta - 2)h_1 \end{aligned}$$

where $\delta = \frac{v^{-1}-v}{z} + 1$ as defined in Chapter 1.

We substitute these expressions into each of the settings to obtain the following:

$$\begin{aligned}
P_2(\sigma_1 Y_2 + Y_2\sigma_1^{-1}) &= P_2((z^2+4)e + z^2(\delta-2)h_1) \\
&= (z^2+4)P_2(e) + z^2(\delta-2)P_2(h_1) \\
&= (z^2+4)a^2 + z^2(\delta-2)a^0 \\
N_2(\sigma_1 Y_2 + Y_2\sigma_1^{-1}) &= (z^2+4)N_2(e) + z^2(\delta-2)N_2(h_1) \\
&= (z^2+4)a^{-2} + z^2(\delta-2)a^0
\end{aligned}$$

Finally, we combine these results with those previously noted to give:

$$\begin{aligned}
l(Y_2) - r(Y_2) &= (r_0(Y_2) - r_1(Y_2)) + (r_1(Y_2) - r_2(Y_2)) \\
&= z(P_2(\sigma_1 Y_2 + Y_2\sigma_1^{-1}) - N_2(\sigma_1 Y_2 + Y_2\sigma_1^{-1})) \\
&= z((z^2+4)a^2 + z^2(\delta-2)a^0) - z((z^2+4)a^{-2} + z^2(\delta-2)a^0) \\
&= z(z^2+4)(a^2 - a^{-2}),
\end{aligned}$$

as required. ∎

We will consider how we can use the main skein relations on expressions of the form $r_k(X) - r_{k+1}(X)$ in the next section, as this will be the approach that we take in general to begin these calculations.

7.4 A general approach for Y_n

Before beginning the actual calculations for Y_3 it is important that we make explicit an approach that we can take in general for these kinds of calculations, as well as list general relations that are useful now that we are moving to a setting with more than two braid strings.

For the calculations involving Y_2 we took the step of rewriting the expression that we started with as

$$l(Y_2) - r(Y_2) = (r_0(Y_2) - r_1(Y_2)) + (r_1(Y_2) - r_2(Y_2)),$$

which we then applied the main Kauffman skein relation to in order to ultimately allow us to express the diagrams as a sum of elements a^m, $m \in \mathbb{Z}$.

A more general statement can be made along these lines, but in order to do that we must first introduce several more pieces of notation and show how they are equivalent to other objects in the skein of the annulus with two connected boundary points.

Definition

For X, a linear combination of braid words from $\mathbb{B}_n$, we define the settings $r_{k,0}(X)$ and $r_{k,\infty}(X)$ to be similar to the closure of $r_k(X)$ with the difference in the arrangement of the arc connecting the interior boundary point to the exterior boundary point as shown in Figure 7.10.

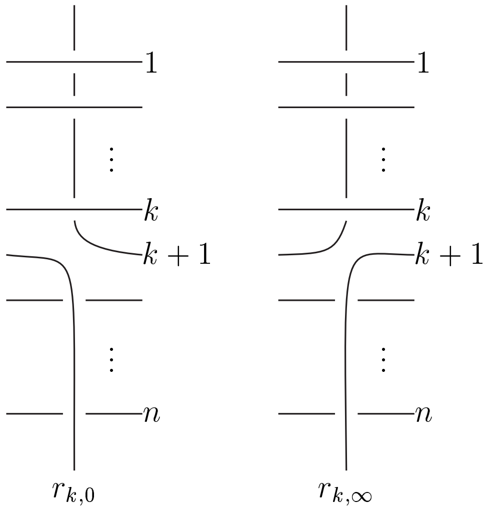


Figure 7.10: Arrangement of arcs near boundary points for $r_{k,0}(X)$ and $r_{k,\infty}(X)$

Lemma 7.3 *For X, a linear combination of braid words from $\mathbb{B}_n$,*

$$\begin{aligned} r_k(X) - r_{k+1}(X) &= zP_n(\sigma_{n-1}\dots\sigma_{k+1}X\sigma_k^{-1}\dots\sigma_1^{-1}) \\ &\quad - zN_n(\sigma_1\dots\sigma_k X\sigma_{k+1}^{-1}\dots\sigma_{n-1}^{-1}) \end{aligned}$$

for $0 \leq k \leq n-1$.

Proof

By the main Kauffman skein relations, we state that

$$r_k(X) - r_{k+1}(X) = z(r_{k,0}(X) - r_{k,\infty}(X)).$$

From considering the diagrams in the annulus it is not difficult to see that

$$\begin{aligned} r_{k,0}(X) &= P_n(\sigma_{n-1}\ldots\sigma_{k+1}X\sigma_k^{-1}\ldots\sigma_1^{-1}) \\ r_{k,\infty}(X) &= N_n(\sigma_1\ldots\sigma_k X\sigma_{k+1}^{-1}\ldots\sigma_{n-1}^{-1}), \end{aligned}$$

which gives the required result. ∎

We obtain an extension to Lemma 7.3, which gives us a good foundation for the problem that we wish to tackle.

Lemma 7.4 *For X, a linear combination of braid words from $\mathbb{B}_n$,*

$$\begin{aligned} l(X) - r(X) &= zP_n(\sigma_{n-1}\ldots\sigma_1 X + \sigma_{n-1}\ldots\sigma_2 X\sigma_1^{-1} + \ldots + X\sigma_{n-1}^{-1}\ldots\sigma_1^{-1}) \\ &\quad - zN_n(\sigma_1\ldots\sigma_{n-1}X + \sigma_1\ldots\sigma_{n-2}X\sigma_{n-1}^{-1} + \ldots + X\sigma_1^{-1}\ldots\sigma_{n-1}^{-1}) \end{aligned}$$

Proof
To begin with state

$$\begin{aligned} l(X) - r(X) &= r_0(X) - r_n(X) \\ &= (r_0(X) - r_1(X)) + (r_1(X) - r_2(X)) + \ldots + (r_{n-1}(X) - r_n(X)). \end{aligned}$$

By Lemma 7.3 we can express every $r_i(X) - r_{i+1}(X)$ as an expression in terms of diagrams in the settings P_n and N_n multiplied by z. We work over all i from 0 to $n-1$, and so

$$\begin{aligned} l(X) - r(X) &= \sum_{i=0}^{n-1}(r_i(X) - r_{i+1}(X)) \\ &= z\sum_{i=0}^{n-1}(r_{i,0}(X) - r_{i,\infty}(X)) \\ &= z\sum_{i=0}^{n-1}(P_n(\sigma_{n-1}\ldots\sigma_{i+1}X\sigma_i^{-1}\ldots\sigma_1^{-1}) - N_n(\sigma_1\ldots\sigma_i X\sigma_{i+1}^{-1}\ldots\sigma_{n-1}^{-1})) \end{aligned}$$

giving the required result. ∎

Lemma 7.4 is the starting point for showing the desired result for Y_3; I believe it is a good starting point for this type of calculation in general.

7.4.1 Summary of relations

Before we proceed with the calculations for Y_3 we summarise relations that can be observed in the contexts we have discussed. Some of these are derived from purely algebraic considerations, while others are obtained directly from how we can manipulate the geometric objects in the annulus settings.

$$\begin{array}{lrcl}
\text{(K1)} & \sigma_i - \sigma_i^{-1} & = & z(e - h_i) \\
\text{(K2)} & \sigma_i^{\pm 1} h_i & = & v^{\pm 1} h_i \\
\text{(R1)} & \sigma_i^2 & = & z\sigma_i - vzh_i + e \\
\text{(R2)} & \sigma_i^{-2} & = & -z\sigma_i^{-1} + v^{-1}zh_i + e \\
\text{(R3)} & \sigma_i^2 + \sigma_i^{-2} & = & (z^2+2)e + z^2(\delta - 2)h_i \\
\text{(R4)} & \sigma_i^{\pm 1} h_{i+1} \sigma_i^{\pm 1} & = & \sigma_{i+1}^{\mp 1} h_i \sigma_{i+1}^{\mp 1} \\
\text{(R5)} & h_i^2 & = & \delta h_i \\
\text{(R6a)} & \sigma_{i+1}^{\pm 1}\sigma_i^{\pm 1}\sigma_{i+1}^{\mp 1}\sigma_i^{\mp 1} & = & \sigma_i^{\mp 1}\sigma_{i+1}^{\pm 1} \\
\text{(R6b)} & \sigma_i^{\pm 1}\sigma_{i+1}^{\pm 1}\sigma_i^{\mp 1}\sigma_{i+1}^{\mp 1} & = & \sigma_{i+1}^{\mp 1}\sigma_i^{\pm 1} \\
\text{(H1a)} & h_{i+1}h_ih_{i+1} & = & h_{i+1} \\
\text{(H1b)} & h_ih_{i+1}h_i & = & h_i
\end{array}$$

In the result of Lemma 7.3 we implicitly used the following result, whose proof can be observed simply from considering diagrams in the relevant setting.

Lemma 7.5 *Take a linear combination of braid words X from $\mathbb{B}_n$ for $n \geq 3$. Then in $P_n(X)$ we can remove $\sigma_k^{\pm 1}$ or h_k for $1 \leq k \leq n-2$ at the start of a word at the expense of adding, respectively, $\sigma_{k+1}^{\pm 1}$ or h_{k+1} to the end of a word. Similarly in $N_n(X)$ we can remove $\sigma_k^{\pm 1}$ or h_k for $1 \leq k \leq n-2$ at the end of a word and in its place add $\sigma_{k+1}^{\pm 1}$ or h_{k+1} to the start of the word.*

Effectively we are sliding these crossings or turnbacks around the annulus as the setting allows; in the calculations that follow we will refer to applications of Lemma 7.5 as using *slide moves.*

7.5 Calculations for Y_3

Theorem 7.6 $l(Y_3) - r(Y_3) = z(z^2+3)^2(a^3 - a^{-3}) + z^3(\delta - 2)(a - a^{-1})$.

We take a first step and state by Lemma 7.4

$$\begin{aligned} l(Y_3) - r(Y_3) &= zP_3(\sigma_2\sigma_1 Y_3 + \sigma_2 Y_3 {\sigma_1}^{-1} + Y_3 {\sigma_2}^{-1}{\sigma_1}^{-1}) \\ &\quad -zN_3(\sigma_1\sigma_2 Y_3 + \sigma_1 Y_3 {\sigma_2}^{-1} + Y_3 {\sigma_1}^{-1}{\sigma_2}^{-1}). \end{aligned}$$

For ease of calculation we will calculate these two terms separately, and then bring them together afterwards.

7.5.1 $P_3(\sigma_2\sigma_1 Y_3 + \sigma_2 Y_3 {\sigma_1}^{-1} + Y_3 {\sigma_2}^{-1}{\sigma_1}^{-1})$

Denote $\sigma_2\sigma_1 Y_3 + \sigma_2 Y_3 {\sigma_1}^{-1} + Y_3 {\sigma_2}^{-1}{\sigma_1}^{-1}$ as the following for ease of reference:

$$ {Y_3}^{+} := \sigma_2\sigma_1 Y_3 + \sigma_2 Y_3 {\sigma_1}^{-1} + Y_3 {\sigma_2}^{-1}{\sigma_1}^{-1}. $$

Lemma 7.7 $P_3(Y_3^{+}) = (z^4 + 6z^2 + 9)a^3 + 2z^2(\delta - 3)a^1 + z^2(2\delta - 6 - z^2)a^{-1} + z^2(v^{-1}r(Y_1) + vl(Y_1))$.

Proof
In the first instance we perform skein relations not specific to the setting P_3, i.e., we do not perform slide moves as described by Lemma 7.5.

We begin the evaluation by expanding the expression for ${Y_3}^{+}$ in terms of a sum of braid words, and use relations to simplify any expressions which can obviously be simplified. Hence

$$\begin{aligned} {Y_3}^{+} &= \sigma_2\sigma_1 Y_3 + \sigma_2 Y_3 {\sigma_1}^{-1} + Y_3 {\sigma_2}^{-1}{\sigma_1}^{-1} \\ &= \sigma_2\sigma_1\sigma_2\sigma_1 + \sigma_2\sigma_1{\sigma_2}^{-1}\sigma_1 + \sigma_2\sigma_1{\sigma_2}^{-1}{\sigma_1}^{-1} \\ &\quad + \sigma_2\sigma_2\sigma_1{\sigma_1}^{-1} + \sigma_2{\sigma_2}^{-1}\sigma_1{\sigma_1}^{-1} + \sigma_2{\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_1}^{-1} \\ &\quad + \sigma_2\sigma_1{\sigma_2}^{-1}{\sigma_1}^{-1} + {\sigma_2}^{-1}\sigma_1{\sigma_2}^{-1}{\sigma_1}^{-1} + {\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} \\ &= {\sigma_2}^2 + {\sigma_1}^{-2} + e + 2\sigma_1{\sigma_2}^{-1} + \sigma_2\sigma_1\sigma_2\sigma_1 \\ &\quad + {\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} + \sigma_2\sigma_1{\sigma_2}^{-1}\sigma_1 + {\sigma_2}^{-1}\sigma_1{\sigma_2}^{-1}{\sigma_1}^{-1} \end{aligned}$$

Our initial method is to take words of length four and use skein relations to express them as linear combinations of words of length three or smaller. We may have to express them as words of length four involving h_i as an intermediate step. We tackle words in terms of generators and inverses only first, and then consider any words of length four with elements h_i.

We first find expressions for $\sigma_2\sigma_1{\sigma_2}^{-1}\sigma_1$ and ${\sigma_2}^{-1}\sigma_1{\sigma_2}^{-1}{\sigma_1}^{-1}$, as we can then combine these with other words in our expression.

$$\begin{aligned}
\sigma_2\sigma_1{\sigma_2}^{-1}\sigma_1 &= \sigma_2\sigma_1(\sigma_2 - ze + zh_2)\sigma_1 \\
&= \sigma_2\sigma_1\sigma_2\sigma_1 - z\sigma_2{\sigma_1}^2 + z\sigma_2\sigma_1 h_2\sigma_1 \\
&= \sigma_2\sigma_1\sigma_2\sigma_1 - z\sigma_2{\sigma_1}^2 + zh_1{\sigma_2}^{-1} \\
{\sigma_2}^{-1}\sigma_1{\sigma_2}^{-1}{\sigma_1}^{-1} &= {\sigma_2}^{-1}({\sigma_1}^{-1} + ze - zh_1){\sigma_2}^{-1}{\sigma_1}^{-1} \\
&= {\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} + z{\sigma_2}^{-2}{\sigma_1}^{-1} - z{\sigma_2}^{-1}h_1{\sigma_2}^{-1}{\sigma_1}^{-1} \\
&= {\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} + z{\sigma_2}^{-2}{\sigma_1}^{-1} - z\sigma_1 h_2
\end{aligned}$$

Substituting these in to our expression for ${Y_3}^+$ gives

$$\begin{aligned}
{Y_3}^+ \;=\;& {\sigma_2}^2 + {\sigma_1}^{-2} + e + 2\sigma_1{\sigma_2}^{-1} + 2\sigma_2\sigma_1\sigma_2\sigma_1 + 2{\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} \\
&+ z({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) + z(h_1\sigma_2^{-1} - \sigma_1 h_2).
\end{aligned}$$

Two terms that we need to evaluate now are $\sigma_2\sigma_1\sigma_2\sigma_1$ and ${\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1}$. We use a combination of skein relations and relations equivalent to the Type III Reidemeister move to simplify these expressions.

$$\begin{aligned}
\sigma_2\sigma_1\sigma_2\sigma_1 &= ({\sigma_2}^{-1} + z - zh_2)\sigma_1\sigma_2\sigma_1 \\
&= {\sigma_2}^{-1}\sigma_2\sigma_1\sigma_2 + z\sigma_1\sigma_2\sigma_1 - zh_2\sigma_2\sigma_1\sigma_2 \\
&= \sigma_1\sigma_2 + z\sigma_1\sigma_2\sigma_1 - vzh_2\sigma_1\sigma_2 \\
{\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} &= (\sigma_2 - z + zh_2){\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} \\
&= \sigma_2{\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1} - z{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} + zh_2{\sigma_2}^{-1}{\sigma_1}^{-1}{\sigma_2}^{-1} \\
&= {\sigma_1}^{-1}{\sigma_2}^{-1} - z{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} + v^{-1}zh_2{\sigma_1}^{-1}{\sigma_2}^{-1}
\end{aligned}$$

Substituting this in to the expression for ${Y_3}^+$ we see the following:

$$\begin{aligned}
{Y_3}^+ &= {\sigma_2}^2 + {\sigma_1}^{-2} + e + 2\sigma_1{\sigma_2}^{-1} + 2(\sigma_1\sigma_2 + z\sigma_1\sigma_2\sigma_1 - vzh_2\sigma_1\sigma_2) \\
&\quad +2({\sigma_1}^{-1}{\sigma_2}^{-1} - z{\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1} + v^{-1}zh_2{\sigma_1}^{-1}{\sigma_2}^{-1}) \\
&\quad + z({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) + z(h_1\sigma_2^{-1} - \sigma_1 h_2) \\
&= {\sigma_2}^2 + {\sigma_1}^{-2} + e + 2(\sigma_1{\sigma_2}^{-1} + \sigma_1\sigma_2 + {\sigma_1}^{-1}{\sigma_2}^{-1}) \\
&\quad +2z(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1}) + z({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) \\
&\quad +2z(v^{-1}h_2{\sigma_1}^{-1}{\sigma_2}^{-1} - vh_2\sigma_1\sigma_2) + z(h_1\sigma_2^{-1} - \sigma_1 h_2)
\end{aligned}$$

We have reduced all words of length four to an expression of words of length three or less. Before we repeat the process, eliminating all words of length three and expressing them in terms of words of length two or less, we will consider what we can say so far about $P_3(Y_3^+)$.

We begin by considering several of the terms in our expression for Y_3^+, and use slide moves to simplify them in the P_3 setting.

$$\begin{aligned}
P_3(\sigma_1{\sigma_2}^{-1} + \sigma_1\sigma_2 + {\sigma_1}^{-1}{\sigma_2}^{-1}) &= P_3(e + \sigma_1^2 + \sigma_1^{-2}) \\
P_3(v^{-1}h_2{\sigma_1}^{-1}{\sigma_2}^{-1} - vh_2\sigma_1\sigma_2) &= P_3(v^{-1}{\sigma_1}^{-1}h_2{\sigma_1}^{-1} - v\sigma_1h_2\sigma_1) \\
P_3(h_1\sigma_2^{-1} - \sigma_1h_2) &= P_3((v^{-1} - v)h_1)
\end{aligned}$$

Then

$$\begin{aligned}
P_3(Y_3^+) &= 3P_3(e + \sigma_1^2 + \sigma_1^{-2}) + z^2(\delta - 1)P_3(h_1) + 2zP_3(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1}) \\
&\quad +zP_3({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) + 2zP_3(v^{-1}{\sigma_1}^{-1}h_2{\sigma_1}^{-1} - v\sigma_1h_2\sigma_1)
\end{aligned}$$

By one of our earlier results we can express $\sigma_1^2 + \sigma_1^{-2}$ as a linear combination of e and h_1, and in turn we can evaluate these as linear combinations of elements of the form a^m, $m \in \mathbb{Z}$; however we postpone doing that for now as we will find other elements to add to these.

We turn our attention to the words of length three that we have in our expression for $P_3(Y_3^+)$. We stay in the setting P_3 to take advantage of slide moves.

We begin by examining the expression $P_3(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1})$. We use the skein relations in a manner that mirrors our earlier proof for Lemma 7.4.

$$\begin{aligned}
P_3(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1}) &= P_3(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}\sigma_2\sigma_1) \\
&\quad + P_3({\sigma_1}^{-1}\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}\sigma_1) \\
&\quad + P_3({\sigma_1}^{-1}{\sigma_2}^{-1}\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1}) \\
&= P_3((\sigma_1 - {\sigma_1}^{-1})\sigma_2\sigma_1) + P_3({\sigma_1}^{-1}(\sigma_2 - {\sigma_2}^{-1})\sigma_1) \\
&\quad + P_3({\sigma_1}^{-1}{\sigma_2}^{-1}(\sigma_1 - {\sigma_1}^{-1})) \\
&= zP_3(\sigma_2\sigma_1 + {\sigma_1}^{-1}\sigma_1 + {\sigma_1}^{-1}{\sigma_2}^{-1}) \\
&\quad - zP_3(h_1\sigma_2\sigma_1 + {\sigma_1}^{-1}h_2\sigma_1 + {\sigma_1}^{-1}{\sigma_2}^{-1}h_1) \\
&= zP_3(e + {\sigma_1}^{-2} + \sigma_2\sigma_1) \\
&\quad - zP_3(h_1\sigma_2\sigma_1 + \sigma_1 h_2\sigma_1 + {\sigma_1}^{-1}h_2\sigma_1).
\end{aligned}$$

We evaluate $P_3({\sigma_1}^{-1}h_2\sigma_1)$ separately as it suits our purposes to have all the signs of elements in these words to be of the same type.

$$\begin{aligned}
P_3({\sigma_1}^{-1}h_2\sigma_1) &= P_3((\sigma_1 - ze + zh_1)h_2\sigma_1) \\
&= P_3(\sigma_1 h_2\sigma_1 - zh_2\sigma_1 + zh_1h_2\sigma_1).
\end{aligned}$$

Then for $P_3(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1})$ we obtain the following expression:

$$\begin{aligned}
P_3(\sigma_1\sigma_2\sigma_1 - {\sigma_1}^{-1}{\sigma_2}^{-1}{\sigma_1}^{-1}) &= zP_3(e + {\sigma_1}^{-2} + \sigma_2\sigma_1 + zh_2\sigma_1) \\
&\quad - zP_3(2\sigma_1h_2\sigma_1 + h_1\sigma_2\sigma_1 + zh_1h_2\sigma_1).
\end{aligned}$$

Substituting this in to the expression for $P_3({Y_3}^{+})$ gives

$$\begin{aligned}
P_3(Y_3^{+}) &= 3P_3(e + \sigma_1^2 + {\sigma_1}^{-2}) + z^2(\delta - 1)P_3(h_1) + 2z^2P_3(e + {\sigma_1}^{-2} + \sigma_2\sigma_1 + zh_2\sigma_1) \\
&\quad - 2z^2P_3(2\sigma_1h_2\sigma_1 + h_1\sigma_2\sigma_1 + zh_1h_2\sigma_1) \\
&\quad + zP_3({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) + 2zP_3(v^{-1}{\sigma_1}^{-1}h_2{\sigma_1}^{-1} - v\sigma_1h_2\sigma_1)
\end{aligned}$$

We will evaluate ${\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2$ shortly; using quadratic relations previously derived we know that we will obtain words of length two or one. Thus it suits us to now eliminate the remaining words of length three which contain h_i. By considering these words in the setting P_3 we obtain the following values:

$$\begin{aligned} P_3({\sigma_1}^{-1}h_2{\sigma_1}^{-1}) &= a^{-1} & \qquad P_3(\sigma_1 h_2 \sigma_1) &= a^{-1} \\ P_3(h_1h_2\sigma_1) &= v^{-1}P_3(h_1) & \qquad P_3(h_1\sigma_2\sigma_1) &= \delta a^1 \end{aligned}$$

Then

$$\begin{aligned} P_3({Y_3}^+) &= 3P_3(\sigma_1^2 + {\sigma_1}^{-2}) + (2z^2+3)P_3(e) + z^2(\delta - 1)P_3(h_1) - 2v^{-1}z^3P_3(h_1) \\ &+ 2z^2P_3({\sigma_1}^{-2} + \sigma_2\sigma_1 + zh_2\sigma_1) + zP_3({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) \\ &- 2z^2\delta a^1 + 2z^2(\delta - 3)a^{-1} \end{aligned}$$

We reduce the expression $P_3({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2)$ using quadratic relations. We present single crossings as products of diagrams in the annulus.

$$\begin{aligned} P_3({\sigma_2}^{-2}{\sigma_1}^{-1} - \sigma_2{\sigma_1}^2) &= P_3((zv^{-1}h_2 - z{\sigma_2}^{-1} + e){\sigma_1}^{-1}) \\ &\quad - P_3(\sigma_2(z\sigma_1 - zvh_1 + e)) \\ &= zP_3(v^{-1}h_2{\sigma_1}^{-1} + v\sigma_2 h_1) \\ &\quad - zP_3({\sigma_2}^{-1}{\sigma_1}^{-1} + \sigma_2\sigma_1) \\ &\quad + (r(Y_1) - l(Y_1)) \cdot a^2. \end{aligned}$$

Making this substitution we have an expression for $P_3({Y_3}^+)$ that contains words of at most length two.

$$\begin{aligned} P_3({Y_3}^+) &= 3P_3(\sigma_1^2 + {\sigma_1}^{-2}) + (2z^2+3)P_3(e) + z^2(\delta - 1)P_3(h_1) - 2v^{-1}z^3P_3(h_1) \\ &+ 2z^2P_3({\sigma_1}^{-2} + zh_2\sigma_1) + z^2P_3(\sigma_2\sigma_1 - \sigma_2^{-1}\sigma_1^{-1}) \\ &+ z^2P_3(v^{-1}h_2\sigma_1^{-1} + v\sigma_2h_1) \\ &+ z(r(Y_1) - l(Y_1)) \cdot a^2 - 2z^2\delta a^1 + 2z^2(\delta - 3)a^{-1} \end{aligned}$$

We use skein relations again to remove all words of length two. For now we leave the term of $P_3(\sigma_1^2 + {\sigma_1}^{-2})$.

We see the following expression for $P_3(\sigma_2\sigma_1 - {\sigma_2}^{-1}{\sigma_1}^{-1})$ following a similar method to before:

$$\begin{aligned} P_3(\sigma_2\sigma_1 - {\sigma_2}^{-1}{\sigma_1}^{-1}) &= P_3(\sigma_2\sigma_1 - {\sigma_2}^{-1}\sigma_1) + P_3({\sigma_2}^{-1}\sigma_1 - {\sigma_2}^{-1}{\sigma_1}^{-1}) \\ &= zP_3((e-h_2)\sigma_1) + zP_3({\sigma_2}^{-1}(e-h_1))) \\ &= zP_3(\sigma_1 + {\sigma_1}^{-1} - h_2\sigma_1 - {\sigma_2}^{-1}h_1) \\ &= z(l(Y_1) + r(Y_1)) \cdot a^2 - zP_3(h_2\sigma_1 + {\sigma_2}^{-1}h_1) \end{aligned}$$

Applying this with the quadratic relation for ${\sigma_i}^{-2}$, and collecting terms, we obtain the following expression for $P_3({Y_3}^+)$:

$$\begin{aligned} P_3(Y_3^+) &= 3P_3(\sigma_1^2 + {\sigma_1}^{-2}) + (4z^2+3)P_3(e) + z^2(\delta-1)P_3(h_1) \\ &\quad + z^2P_3(v^{-1}h_2\sigma_1^{-1} + v\sigma_2h_1 + zh_2\sigma_1 - z\sigma_2^{-1}h_1) \\ &\quad + (z^3-z)(l(Y_1) - r(Y_1)) \cdot a^2 - 2z^2\delta a^1 + 2z^2(\delta-3)a^{-1} \end{aligned}$$

By considering their diagrams we can evaluate the following words of length two as follows:

$$\begin{aligned} P_3(h_2\sigma_1) &= l(Y_1) = P_3(\sigma_2h_1) \\ P_3(h_2{\sigma_1}^{-1}) &= r(Y_1) = P_3({\sigma_2}^{-1}h_1) \end{aligned}$$

We make these substitutions, along with the quadratic relation for ${\sigma_1}^2 + {\sigma_1}^{-2}$, and recall from Lemma 7.1 that $l(Y_1) - r(Y_1) = z(a - a^{-1})$. Finally, in the setting of P_3 we observe that e evaluates to a^3 and a single h_i evaluates to a^1.

$$\begin{aligned} P_3(Y_3^+) &= 3P_3((z^2+2)e + z^2(\delta-2)h_1) + (4z^2+3)P_3(e) + z^2(\delta-1)P_3(h_1) \\ &\quad + z^2(v^{-1}r(Y_1) + vl(Y_1) + zl(Y_1) - zr(Y_1)) \\ &\quad + (z^3-z)(l(Y_1) - r(Y_1)) \cdot a^2 - 2z^2\delta a^1 + 2z^2(\delta-3)a^{-1} \\ &= (7z^2+9)P_3(e) + z^2(4\delta-7)P_3(h_1) \\ &\quad + z^2(v^{-1}r(Y_1) + vl(Y_1)) + z^4(a^1 - a^{-1}) \\ &\quad + (z^4-z^2)(a^1-a^{-1}) \cdot a^2 - 2z^2\delta a^1 + 2z^2(\delta-3)a^{-1} \\ &= (z^4+6z^2+9)a^3 + 2z^2(\delta-3)a^1 \\ &\quad + z^2(2\delta-6-z^2)a^{-1} + z^2(v^{-1}r(Y_1) + vl(Y_1)). \end{aligned}$$

∎

7.5.2 $N_3(\sigma_1\sigma_2 Y_3 + \sigma_1 Y_3 {\sigma_2}^{-1} + Y_3 {\sigma_1}^{-1}{\sigma_2}^{-1})$

Denote $\sigma_1\sigma_2 Y_3 + \sigma_1 Y_3 {\sigma_2}^{-1} + Y_3 {\sigma_1}^{-1}{\sigma_2}^{-1}$ as the following:

$$ {Y_3}^- := \sigma_1\sigma_2 Y_3 + \sigma_1 Y_3 {\sigma_2}^{-1} + Y_3 {\sigma_1}^{-1}{\sigma_2}^{-1}. $$

Lemma 7.8 $N_3(Y_3^-) = (z^4+6z^2+9)a^{-3} + z^2(3\delta - 8 - z^2)a^{-1} + z^2(\delta-4)a^1 + z^2(v^{-1}r(Y_1) + vl(Y_1))$.

Proof
Omitted for brevity. By a similar method of manipulations to the $P_3(Y_3^+)$ case we obtain the result. Due to the initial form that the expression takes the rearrangement in this case is easier than in the previous case. ∎

7.5.3 Proof of Theorem 7.6

Proof
In Lemmas 7.7 and 7.8 we calculated

$$\begin{aligned} P_3({Y_3}^+) &= (z^4+6z^2+9)a^3 + 2z^2(\delta-3)a^1 \\ &\quad + z^2(2\delta - 6 - z^2)a^{-1} + z^2(v^{-1}r(Y_1) + vl(Y_1)) \\ N_3({Y_3}^-) &= (z^4+6z^2+9)a^{-3} + z^2(3\delta - 8 - z^2)a^{-1} \\ &\quad + z^2(\delta-4)a^1 + z^2(v^{-1}r(Y_1) + vl(Y_1)). \end{aligned}$$

By Lemma 7.4 we know $l(Y_3) - r(Y_3) = z(P_3({Y_3}^+) - N_3({Y_3}^-))$. Then

$$\begin{aligned} P_3({Y_3}^+) - N_3({Y_3}^-) &= (z^4+6z^2+9)(a^3 - a^{-3}) + z^2(2\delta - 6 - \delta + 4)a^1 \\ &\quad + z^2(2\delta - 6 - z^2 - 3\delta + 8 + z^2)a^{-1} \\ &= (z^4+6z^2+9)(a^3 - a^{-3}) + z^2(\delta-2)a^1 + z^2(2-\delta)a^{-1} \\ &= (z^2+3)^2(a^3 - a^{-3}) + z^2(\delta-2)(a^1 - a^{-1}). \end{aligned}$$

Thus

$$\begin{aligned} l(Y_3) - r(Y_3) &= z(P_3(Y_3^+) - N_3(Y_3^-)) \\ &= z(z^2+3)^2(a^3 - a^{-3}) + z^3(\delta-2)(a - a^{-1}), \end{aligned}$$

as required. ∎

7.6 Y_n, $n > 3$

Following calculations for Y_3 a variety of methods were used to calculate the linear combination of diagrams expressing $l(Y_4) - r(Y_4)$, but all of them were ultimately unsuccessful. The problems in resolving these calculations was largely due to human error. For Y_4^+ and Y_4^- we begin with sixteen braid words on four strings, with each word initially being of length six. Some of these can be simplified immediately, but especially for the calculation of $P_4(Y_4^+)$ we find that we have a large number of words and a large number of intermediate steps when using skein relations to reduce the length of braid words.

It seems that for $n > 3$, the number of intermediate terms and steps in the calculation of $P_n(Y_n^+) - N_n(Y_n^-)$ is too great to realistically be achieved by hand. There are too many terms that can occur, and too many steps that must be taken – both of which contribute to the possibility of human error.

A Maple routine adapted from the algorithms of Chapter 3 gave mixed results. Coefficients of equivalent diagrams were collected, and we can be confident that no errors were made to this point. This left the task of having to manually evaluate a large number of terms with the added complication that some terms that we had previously resolved with skein relations (e.g., reducing of $\sigma_2\sigma_1 - \sigma_2^{-1}\sigma_1^{-1}$) now had only one term remaining in the expression.

It is possible that an alternate form of notation might be used to simplify things, although we have not been able to use any so far to great effect.

By performing calculations modulo the turnback relation in the main Kauffman skein relation we were able to eliminate elements containing h_i for calculations of $l(Y_4) - r(Y_4)$; while this did not allow us to make a complete calculation for Y_4, it did allow us to confirm the following coefficient of $(a^4 - a^{-4})$:

$$l(Y_4) - r(Y_4) = (z^6 + 8z^4 + 20z^2 + 16)(a^4 - a^{-4}) \text{ modulo elements of } h_i.$$

We have performed explicit calculations for only a few cases, but there are some indications as to what might occur in general for $l(Y_n) - r(Y_n)$. To close this chapter let us state a few conjectures that we believe to be true, but have not been able to show.

Proposition 7.9 *We can express $l(Y_n) - r(Y_n)$ as a linear combination of annulus diagrams a^m, $m \in \mathbb{Z}$, $-n \leq m \leq n$, with coefficients $c_{n,m}$ from the Kauffman skein of the annulus. I.e.,*

$$l(Y_n) - r(Y_n) = c_{n,n}a^n + c_{n,n-1}a^{n-1} + \ldots + c_{n,-n}a^{-n}.$$

This is clear from the fact that use of the skein relations will not introduce extra arcs in the annulus: action of skein relations gives a linear combination of diagrams with the same or fewer arcs.

In [31], using different notation, it was shown that an element in the Kauffman skein of the annulus on n strings can be written as a linear combination of elements with n strings and elements with $n - 2k$ strings (for $1 \leq k \leq \frac{n}{2}$, $k \in \mathbb{Z}$). This coincides with our results for Y_3 (Theorem 7.6).

Also, from observations in the Homfly skein of the annulus, and our calculations for $l(Y_3) - r(Y_3)$ and $l(Y_4) - r(Y_4)$, we would conjecture that $c_{n,-n} = -c_{n,n}$ for $l(Y_n) - r(Y_n)$ in general. We expect this because the calculations that showed the result in the Y_4 case were modulo the turnback relation, which is diagrammatically the same as the Homfly relation (although without orientation). It is reasonable to expect that the coefficient of a^0 would be 0 for cases of even n.

From all of these observations and expectations, and coupled with the result of Theorem 7.6 showing $c_{3,-1} = -c_{3,1}$ we thus make the following final conjecture.

Conjecture 7.10 *We can express $l(Y_n) - r(Y_n)$ as a linear combination of annulus diagrams a^m, $m \in \mathbb{Z}$, with coefficients $c_{n,m}$ from the Kauffman skein of the annulus such that*

$$l(Y_n) - r(Y_n) = c_{n,n}(a^n - a^{-n}) + c_{n,n-2}(a^{n-2} - a^{-(n-2)}) + \ldots + c_{n,n-2t}(a^{n-2t} - a^{-(n-2t)})$$

where t is the largest integer less than or equal to $\frac{n}{2}$.

Appendix A

Program Code

In this appendix we give annotated code for the Maple programs `SeqIndex`, `k_plait` and `h_plait` that are mentioned in Chapters 3 and 4 and are developed from the material of those chapters.

A.1 SeqIndex

There are points in the programs `k_plait` and `h_plait` where the program searches through the array of k-sequences in order to find a particular sequence to pass coefficients to; these are for the actions of renumbering, rearrangement and multiplication. Without any other considerations this could be a lengthy task as we have to search through a list of $\frac{(2k)!}{2^k}$ elements to find the element that we require.

The program `SeqIndex` is designed to look through the set of k-sequences for a specific k-sequence and then return the index of that sequence to the main program. It does this efficiently firstly by taking advantage of the way that the `permute` command works in Maple in order to create the array of k-sequences, and secondly by exploiting the combinatorics of how Maple orders the list of k-sequences.

We divide the ordered list of k-sequences into k sections of equal size (size $\frac{(2k)!}{2^k k}$), and the first digit of the k-sequence, T, that we wish to locate is enough to tell us which section it is located in. Already we restrict our search to a subset one kth the size of the set of k-sequences.

We know this due to the regularity with which elements are permuted in Maple; because of that regularity we can narrow the portion of the list that we will have to search through even further by comparing the second digit of T with the first. Then, depending on whether or not it is smaller, larger or the same, we can subdivide the list into even smaller sections. When we search in this way we are effectively searching through all of the sequences which have the same two first digits as T.

This is a massive reduction on having to search through the entire list in order to find an element, and also a great reduction on searching through a kth of the set of k-sequences based on the first digit of T alone.

Input for this routine is the the array to be searched through, and the number sequence to be found. Output is the index for the sequence in the array.

```
###SeqIndex###
##A procedure used to boost efficiency in the main
##HOMFLY and Kauffman procedures that I've created
##Works for both HOMFLY and Kauffman with
##no extra modifications for either needed
SeqIndex := proc(Ay,T)
local k, m, a, p, Ix:
##In an effort to make it as flexible as possible,
##the procedure finds the index of the number
##sequence that is required
##The index is then returned to the main program
##where it is used in rearrangement routines
##or for multiplication
##The input is the array of number sequences
##that is being searched, Ay,
##and the desired number sequence, T
```

```
k:=nops(T)/2:    m:=ArrayNumElems(Ay):
##m is obtained from the array that is brought in
##I decided to obtain the value of k by halving
##the number of operands in the number sequence
##that we wish to find, in order to reduce the number
##of arguments the procedure has to take in
Ix:=0:    p:=0:
##Ix will be the index of the element we wish to find
##p is a marker that halts the search once the
##sequence is found
##We have three situations, T[1]=T[2], T[1]<T[2] or T[1]>T[2]
##When the correct index is found Ix is set to
##that value, and the search ends
if T[1]=T[2] then
  while p=0 do
    for a from 1+((T[1]-1)*m/k)
                +(T[1]-1)*2*(2*(k-1))!/(2^(k-1))
    to ((T[1]-1)*m/k)
       +(2*T[1]-1)*(2*(k-1))!/(2^(k-1)) do
      if Ay[a]=T then
        Ix:=a: p:=1:
      fi:
    od:
  od:
elif T[1]<T[2] then
  while p=0 do
    for a from 1+((T[1]-1)*m/k)
                +(2*T[2]-3)*(2*(k-1))!/(2^(k-1))
    to ((T[1]-1)*m/k)
       +(2*T[2]-1)*(2*(k-1))!/(2^(k-1)) do
      if Ay[a]=T then
```

```
        Ix:=a: p:=1:
      fi:
    od:
  od:
elif T[1]>T[2] then
  while p=0 do
    for a from 1+((T[1]-1)*m/k)
                +(T[2]-1)*2*(2*(k-1))!/(2^(k-1))
    to ((T[1]-1)*m/k)
       +2*T[2]*(2*(k-1))!/(2^(k-1)) do
      if Ay[a]=T then
        Ix:=a: p:=1:
      fi:
    od:
  od:
fi:
##The procedure ends by returning Ix
return Ix;
end;
```

A.2 k_plait

The comments for the program are contained within the listing for the program itself.

Input for the program is the width of the plait k, followed by a string of c non-zero integers between $-(k-1)$ and $(k-1)$ indicating the braid word of length c of the plait presentation. Output is the Kauffman polynomial of the plait presentation in variables v and z, collecting coefficients of v against z.

```
###k_plait###
########################################################
#Input for program is k followed by a string of positive
#and negative numbers indicating crossings in the
#plait presentation
########################################################
#Initialisation Part 1
########################################################
#Introduce the permute command outside of
#the main program listing
###
#"permute" allows us to generate the set of
#k-sequences
########################################################
with(combinat,permute):
########################################################
#Initialisation Part 2
########################################################
#We initialise the program and define
#the variables that we will use
########################################################
k_plait := proc()
local a,b,c,d,f,i,j,k,l,m,n,p,r,t,w,x,y,
A,Anow,Anext,B,Bnow,Bnext,C,Y,Ynext,
depth,posn,switch,sm_plus,sm_minus,
mult_temp,close_temp,close_mult,
delta,output:
########################################################
#With the exception of k, the plait number,
#lower case variables are looping variables
#or flags, and occasionally temporary variables
###
#Upper case variables are arrays/lists
#generated by the program
###
#Variables with "names" will be explained
#in commenting in the first instance of their use
########################################################
#Initialisation Part 3
########################################################
#In the final initialisation section we
#create the set of k-sequences and
#the array that stores coefficients
########################################################
k:=args[1]:
Y:=[seq(x[i],i=1..2*k)]:
for a from 1 to 2*k do
  if type(a/2,integer) then
    Y[a]:=a/2:
  else
    Y[a]:=(a+1)/2:
  fi:
od:
##Preceding lines generate the initial
```

```
##sequence that we then permute in
##the following lines in order to give
##the set of k-sequences, which we
##store in arrays, along with an array
##for the coefficients attached to the
##sequences
C:=permute(Y,2*k):
A:=Array(1..nops(C)):
B:=Array(1..nops(C)):
m:=ArrayNumElems(A):
for a from 1 to m do A[a]:=C[a]: B[a]:=0: od:
B[1]:=1:
C:='C':
########################################################
#Initialisation complete
########################################################
#Start of the real mechanisms of the program
########################################################
for n from 2 to nargs do      #START OF MAIN LOOP
  i:=args[n]:                 #Crossing from the plait
########################################################
#START OF REARRANGEMENT/RENUMBERING LOOP
########################################################
  for r from k to 2 by -1 do
    for j from 1 to m do      #Looping through all A,B
      if i>0 then             #Case for positive crossings
        if B[j]<>0 and A[j][i+1]=r and A[j][i]<r then
          depth:=[0,0,0,0]: posn:=[0,0,0,0]: w:=1:
      ##If rearrangement is needed for a
      ##particular k-sequence then these
      ##lines obtain the information
```

```
##that allow us to determine the
##k-sequence that coefficients
##will be passed to.
####
##depth and posn store the
##information for the adjacent
##arcs that we are performing
##skein relations on
####
##If rearrangement is needed for a
##k-sequence to be compatible
##then we always need the
##following lines to get key
##information
    while w<5 do
      for c from 1 to 2*k do
        if A[j][c]=r-1 or A[j][c]=r then
          depth[w]:=A[j][c]: posn[w]:=c: w:=w+1:
        fi:
      od:
    od:
##For rearrangement or renumbering
##we always pass coefficients to a
##k-sequence represented by the
##object 'switch'
    switch:=A[j]:
    for f from 1 to 2*k do
      if A[j][f]=r-1 then
        switch[f]:=r:
      elif A[j][f]=r then
        switch[f]:=r-1:
```

```
      fi:
    od:
##If we need more than the action
##of a renumbering operation we need
##the following series of steps to
##determine the other k-sequences
##that coefficients are passed to
####
##sm_plus and sm_minus are
##are the two k-sequences that
##represent the smoothings in
##the main skein relations
####
##sm_plus and sm_minus are
##determined by the value of
##depth[1], along with various
##posn values
    if depth[1]=depth[3] then
      sm_plus:=A[j]: sm_minus:=A[j]:
      if depth[1]=r then
        if posn[1]=i+1 then
          sm_plus[posn[1]]:=r-1:  sm_plus[posn[2]]:=r-1:
          sm_plus[posn[3]]:=r:    sm_plus[posn[4]]:=r:
          sm_minus[posn[1]]:=r-1: sm_minus[posn[2]]:=r:
          sm_minus[posn[3]]:=r:   sm_minus[posn[4]]:=r-1:
        else
          sm_plus[posn[1]]:=r:    sm_plus[posn[2]]:=r:
          sm_plus[posn[3]]:=r-1:  sm_plus[posn[4]]:=r-1:
          sm_minus[posn[1]]:=r:   sm_minus[posn[2]]:=r-1:
          sm_minus[posn[3]]:=r-1: sm_minus[posn[4]]:=r:
        fi:
```

```
     elif depth[1]=r-1 then
       if posn[2]=i+1 then
         sm_plus[posn[1]]:=r:    sm_plus[posn[2]]:=r-1:
         sm_plus[posn[3]]:=r-1:  sm_plus[posn[4]]:=r:
         sm_minus[posn[1]]:=r-1: sm_minus[posn[2]]:=r-1:
         sm_minus[posn[3]]:=r:   sm_minus[posn[4]]:=r:
       else
         sm_plus[posn[1]]:=r-1:  sm_plus[posn[2]]:=r:
         sm_plus[posn[3]]:=r:    sm_plus[posn[4]]:=r-1:
         sm_minus[posn[1]]:=r:   sm_minus[posn[2]]:=r:
         sm_minus[posn[3]]:=r-1: sm_minus[posn[4]]:=r-1:
       fi:
     fi:
   fi:
##Having determined the k-sequences that
##we have to pass coefficients to, we now
##have the routines that move the coefficients
####
##SeqIndex is a called program that finds
##the index of a required k-sequence
   y:=SeqIndex(A,switch):
   B[y]:=simplify(B[y]+B[j]):
   if depth[1]=depth[3] then
     y:=SeqIndex(A,sm_plus):
     B[y]:=simplify(B[y]+z*B[j]):
     y:=SeqIndex(A,sm_minus):
     B[y]:=simplify(B[y]-z*B[j]):
   fi:
   B[j]:=0:
##Delete coefficient after rearrangement
 fi:
```

```
elif i<0 then   #Case for negative crossings
  if B[j]<>0 and A[j][abs(i)]=r
             and A[j][abs(i)+1]<r then
    depth:=[0,0,0,0]: posn:=[0,0,0,0]: w:=1:
    while w<5 do
      for c from 1 to 2*k do
        if A[j][c]=r-1 or A[j][c]=r then
          depth[w]:=A[j][c]: posn[w]:=c: w:=w+1:
        fi:
      od:
    od:
 ##In this section we have similar
 ##pieces of code to previously; these
 ##deal with the case when we need
 ##to ensure compatibility for an
 ##inverse
    switch:=A[j]:
    for f from 1 to 2*k do
      if A[j][f]=r-1 then
        switch[f]:=r:
      elif A[j][f]=r then
        switch[f]:=r-1:
      fi:
    od:
    if depth[1]=depth[3] then
      sm_plus:=A[j]: sm_minus:=A[j]:
      if depth[1]=r-1 then
        if posn[4]=abs(i) then
          sm_plus[posn[1]]:=r-1:  sm_plus[posn[2]]:=r:
          sm_plus[posn[3]]:=r:    sm_plus[posn[4]]:=r-1:
          sm_minus[posn[1]]:=r:   sm_minus[posn[2]]:=r:
```

```
        sm_minus[posn[3]]:=r-1: sm_minus[posn[4]]:=r-1:
      else
        sm_plus[posn[1]]:=r:    sm_plus[posn[2]]:=r-1:
        sm_plus[posn[3]]:=r-1:  sm_plus[posn[4]]:=r:
        sm_minus[posn[1]]:=r-1: sm_minus[posn[2]]:=r-1:
        sm_minus[posn[3]]:=r:   sm_minus[posn[4]]:=r:
      fi:
    elif depth[1]=r then
      if posn[1]=abs(i) then
        sm_plus[posn[1]]:=r-1:  sm_plus[posn[2]]:=r-1:
        sm_plus[posn[3]]:=r:    sm_plus[posn[4]]:=r:
        sm_minus[posn[1]]:=r-1: sm_minus[posn[2]]:=r:
        sm_minus[posn[3]]:=r:   sm_minus[posn[4]]:=r-1:
      else
        sm_plus[posn[1]]:=r:    sm_plus[posn[2]]:=r:
        sm_plus[posn[3]]:=r-1:  sm_plus[posn[4]]:=r-1:
        sm_minus[posn[1]]:=r:   sm_minus[posn[2]]:=r-1:
        sm_minus[posn[3]]:=r-1: sm_minus[posn[4]]:=r:
      fi:
    fi:
  fi:
  y:=SeqIndex(A,switch):
  B[y]:=simplify(B[y]+B[j]):
  if depth[1]=depth[3] then
    y:=SeqIndex(A,sm_plus):
    B[y]:=simplify(B[y]+z*B[j]):
    y:=SeqIndex(A,sm_minus):
    B[y]:=simplify(B[y]-z*B[j]):
  fi:
  B[j]:=0:
fi:
      fi:            #End of routine for negative crossings
    od:                 #End of loop through A,B
  od:
########################################################
#END OF REARRANGEMENT/RENUMBERING LOOP
########################################################
#By this point in the algorithm, the array
#of coefficients has been rearranged so
#that the only k-sequences which can
#have non-zero coefficients are those
#which are compatible with the generator
#or inverse
###
#This is achieved after k-1 passes of the
#set of k-sequences
########################################################
#MULTIPLICATION PROCEDURE
########################################################
#This is a much shorter procedure, there
#is much less work to do in terms
#of searching through the arrays; we have
#two slightly different routines depending
#on whether or not we have a positive or
#negative crossing
########################################################
  if i>0 then
  ##Generator
    for t from 1 to m do
      if B[t]<>0 and (A[t][abs(i)]>=A[t][abs(i)+1]) then
        mult_temp:=A[t]:
        mult_temp[abs(i)]:=A[t][abs(i)+1]:
```

```
        mult_temp[abs(i)+1]:=A[t][abs(i)]:
        if mult_temp=A[t] then
          B[t]:=v*B[t]:
        else
          y:=SeqIndex(A,mult_temp):
          B[y]:=B[t]:
          B[t]:=0:
        fi:
      fi:
    od:
  elif i<0 then
  ##Inverse
    for t from 1 to m do
      if B[t]<>0 and (A[t][abs(i)]<=A[t][abs(i)+1]) then
        mult_temp:=A[t]:
        mult_temp[abs(i)]:=A[t][abs(i)+1]:
        mult_temp[abs(i)+1]:=A[t][abs(i)]:
        if mult_temp=A[t] then
          B[t]:=v^(-1)*B[t]:
        else
          y:=SeqIndex(A,mult_temp):
          B[y]:=B[t]:
          B[t]:=0:
        fi:
      fi:
    od:
  fi:
od:
########################################################
#END OF MAIN REARRANGEMENT AND MULTIPLICATION LOOP
########################################################
```

```
#CLOSURE ROUTINE
#######################################################
#After all of the multiplications are complete
#we must close off each k-sequence because of
#the 'cups'
###
#We perform the closure one 'cup' at a time.
#######################################################
#k is passed in, but it is only used to give a
#value to the first loop which controls the
#overall process and how many times
#it is repeated
#######################################################
#INITIALISING CLOSURE PROCEDURE
#######################################################
Anow:=A: Bnow:=B:
A:='A': B:='B': m:='m':
m:=ArrayNumElems(Anow):
delta:=1+(v^(-1)-v)/z:
##Define delta, value of disjoint unknot
#######################################################
#START CLOSURE LOOP
#######################################################
for l from k to 2 by -1 do
  if (nops(Y)/2)>2 then
##Don't need rearrangement for closure
##of 2-sequence
####
##Start closure rearrangement/renumbering
    for r from (nops(Y)/2) to 3 by -1 do
      for j from 1 to m do
```

```
if (Bnow[j]<>0) and
   ((Anow[j][1]=r and Anow[j][2]<(r-1))
       or (Anow[j][2]=r and Anow[j][1]<(r-1))) then
  depth:=[0,0,0,0]: posn:=[0,0,0,0]: w:=1:
  while w<5 do
    for c from 1 to nops(Y) do
      if Anow[j][c]=r or Anow[j][c]=r-1 then
        depth[w]:=Anow[j][c]: posn[w]:=c: w:=w+1:
      fi:
    od:
  od:
##As before if renumbering or
##rearrangement is required
##we always need a k-sequence
##where the numbers r and r-1
##are interchanged
  switch:=Anow[j]:
  for i from 1 to nops(Y) do
    if Anow[j][i]=r then
      switch[i]:=r-1:
    elif Anow[j][i]=r-1 then
      switch[i]:=r:
    fi:
  od:
  if depth[1]=depth[3] then
##Only one set of rearrangements
##required.
##Neighbouring arcs will always be
##of the form [r,r-1,r,r-1] if we
##need to rearrange
    sm_plus:=Anow[j]: sm_minus:=Anow[j]:
          sm_plus[posn[1]]:=r-1:  sm_plus[posn[2]]:=r-1:
          sm_plus[posn[3]]:=r:    sm_plus[posn[4]]:=r:
          sm_minus[posn[1]]:=r-1: sm_minus[posn[2]]:=r:
          sm_minus[posn[3]]:=r:   sm_minus[posn[4]]:=r-1:
        fi:
        y:=SeqIndex(Anow,switch):
        Bnow[y]:=simplify(Bnow[y]+Bnow[j]):
        if depth[1]=depth[3] then
          y:=SeqIndex(Anow,sm_plus):
          Bnow[y]:=simplify(Bnow[y]+z*Bnow[j]):
          y:=SeqIndex(Anow,sm_minus):
          Bnow[y]:=simplify(Bnow[y]-z*Bnow[j]):
        fi:
        Bnow[j]:=0:
      fi:
    od:
  od:
fi:
##Now all sequences are closure-compatible
####
##Need to initialise variables that will be used
##for the next level of closure, i.e., to pass
##to sequences with one less arc
  Ynext:=Y[1..(nops(Y)-2)]:
  ##Ynext is the base generator string
  C:=permute(Ynext,nops(Ynext)):
  Anext:=Array(1..nops(C)):
  Bnext:=Array(1..nops(C)):
  ##Anext is the set of sequences that
  ##coefficients will be passed to
  ####
  ##Bnext stores corresponding
  ##coefficients for Anext
  for a from 1 to nops(C) do
    Anext[a]:=C[a]:
    Bnext[a]:=0:
  od:
  C:='C':
##Perform closure
  for j from 1 to m do
    if Bnow[j]<>0 then
      close_temp:=Anow[j][3..nops(Y)]:
      ##close_temp is the sequence
      ##that will result from the action
      ##of closure
      close_mult:=0:
      if Anow[j][1]=Anow[j][2] then
        close_mult:=delta*Bnow[j]:
      else
##The following determines the
##multiplier that is carried through
        depth:=[Anow[j][1],Anow[j][2],0,0]: w:=3:
        while w<5 do
          for b from 3 to nops(Y) do
            if (Anow[j][b]=Anow[j][1])
               or (Anow[j][b]=Anow[j][2]) then
              depth[w]:=Anow[j][b]: w:=w+1:
            fi:
          od:
        od:
        if depth[2]=depth[3] then
          close_mult:=1*Bnow[j]:
```

```
        elif depth[2]<depth[3] then
          close_mult:=(1/v)*Bnow[j]:
        elif depth[2]>depth[3] then
          close_mult:=v*Bnow[j]:
        fi:
      fi:
      ##close_mult, by this point
      ##is the coefficient that is
      ##passed to the next stage
      ##accounting for any multiplier
      for i from 1 to nops(Ynext) do
        if close_temp[i]>min(Anow[j][1],Anow[j][2]) then
          close_temp[i]:=close_temp[i]-1:
        fi:
      od:
      y:=SeqIndex(Anext,close_temp):
      Bnext[y]:=simplify(Bnext[y]+close_mult):
    fi:
  od:
########################################################
#END CLOSURE
########################################################
##We have to initialise Anow, Bnow,
##Y and m for the next loop
  Anow:='Anow': Bnow:='Bnow':
  Anow:=Anext: Bnow:=Bnext:
  m:=ArrayNumElems(Anow):
  Anext:='Anext': Bnext:='Bnext':
  Y:=Ynext:
########################################################
#END CLOSURE
```

```
########################################################
#END MAIN PROGRAM
########################################################
od:
########################################################
#FINAL OUTPUT STAGE
########################################################
output:=Bnow[1]:
##The polynomial of the k-plait presentation
##has been calculated, and we output this
##with coefficients of v on z
output:=collect(expand(output),z):
end;
```

A.3 h_plait

The comments for the program are contained within the listing for the program itself.

Input for the program is the width of the plait k, followed by a string of c non-zero integers between $-(k-1)$ and $(k-1)$ indicating the braid word of length c of the plait presentation. Output is the Homfly polynomial of the plait presentation in variables v and z, collecting coefficients of v against z.

As stated in Chapter 4 and the program listing, this implementation requires that the braid word given respect an initial orientation sequence of $(-1, 1, -1, 1, ..., -1, 1)$. If this is not the case then there will most likely be serious error in any calculations; an implementation could be written so that the initial orientation sequence is a value that is taken from input.

```
###h_plait###
########################################################
#IMPORTANT NOTE:
#Input for program is k followed by a
#string of positive and negative numbers
#indicating crossings in an undirected
#braid presentation (ie, monotonic
#but with no orientation).
###
#Orientation is done so that the initial
#tangle with coefficient 1 has
#orientation (-1,1,-1,1,...,-1,1).
###
#If the presentation is not arranged as such,
#then the program will not run correctly -
#essentially the orientation will not be
#consistent throughout - and errors will most
#certainly occur.
```

```
########################################################
#Initialisation Part 1
########################################################
#Introduce the permute command outside of
#the main program listing
###
#"permute" allows us to generate the set of
#k-sequences
########################################################
with(combinat,permute):
########################################################
#Initialisation Part 2
########################################################
#We initialise the program and define
#the variables that we will use
########################################################
with(combinat,permute):
h_plait := proc()
local a,b,c,f,i,j,k,m,n,p,r,t,w,y,mu2,
A,Anow,Anext,B,Bnow,Bnext,C,Y,Ynext,
S,S1,output,delta,close_temp,
close_mult,mult_temp,switch,
smooth,depth,posn,sign:
########################################################
#With the exception of k, the plait number,
#lower case variables are looping variables
#or flags, and occasionally temporary variables
###
#Upper case variables are arrays/lists
#generated by the program
###
```

```
#Variables with "names" will be explained
#in commenting in the first instance of their use
########################################################
#Initialisation Part 3
########################################################
#In the final initialisation section we
#create the set of k-sequences and
#the array that stores coefficients
###
#We also have to create the
#sequence which stores the
#orientation information for
#the k-sequences
########################################################
k:=args[1]:
Y:=[seq(x[i],i=1..2*k)]:
S:=[seq(x[i],i=1..2*k)]:
for a from 1 to 2*k do
  if type(a/2,integer) then
    Y[a]:=a/2: S[a]:=1:
  else
    Y[a]:=(a+1)/2: S[a]:=-1:
  fi:
od:
#The preceding lines generate the
#initial sequence whose entries are
#permuted to give the set of k-sequences.
###
#We also create the list which holds
#the orientation information of the
#non-zero coefficient k-sequences
```

```
C:=permute(Y,2*k):
A:=Array(1..nops(C)):
B:=Array(1..nops(C)):
m:=ArrayNumElems(A):
for a from 1 to m do
  A[a]:=C[a]: B[a]:=0:
od:
B[1]:=1:
C:='C':
#The preceding lines complete the
#initialisation.
###
#The set of k-sequences is created
#from the permutation, and arrays
#are set up to hold these and the
#coefficients.
###
#We devalue C, so that memory
#is not being taken up by this
#during the program's operation.
########################################################
#Initialisation complete
########################################################
#Start of the main mechanisms of the program
########################################################
for n from 2 to nargs do        #START OF MAIN LOOP
  i:=args[n]:                   #Crossing from the plait
########################################################
#START OF REARRANGEMENT/RENUMBERING LOOP
########################################################
  for r from k to 2 by -1 do
for j from 1 to m do    #Looping through all A,B
  if i>0 then           #Case for positive crossings
    if B[j]<>0 and A[j][i+1]=r and A[j][i]<r then
      depth:=[0,0,0,0]: posn:=[0,0,0,0]:
      w:=1: mu2:=0:
  ##If rearrangement is needed for a
  ##particular k-sequence then these lines
  ##obtain the information that allow us to
  ##determine the k-sequence(s) that
  ##coefficients will be passed to.
      while w<5 do
        for c from 1 to 2*k do
          if A[j][c]=r-1 or A[j][c]=r then
            depth[w]:=A[j][c]: posn[w]:=c: w:=w+1:
          fi:
        od:
      od:
  ##If rearrangement or renumbering
  ##is needed we always pass coefficients
  ##to a k-sequence represented by
  ##the variable 'switch'
      switch:=A[j]:
      for f from 1 to 2*k do
        if A[j][f]=r-1 then
          switch[f]:=r:
        elif A[j][f]=r then
          switch[f]:=r-1:
        fi:
      od:
  ##If rearrangement for a sequence is
  ##more than the action of a renumbering
```

```
##operation then we need the following
##long series of lines to determine the
##other sequence that coefficients
##are passed to.
    if depth[1]=depth[3] then
      smooth:=A[j]:
      sign:=[S[posn[1]],S[posn[2]]]:
      if depth[1]=r-1 then
##smooth is determined by several factors:
##the value of depth[1], the sequence 'sign'
##which is constructed from the orientation
##information stored in S, and various posn
##values which give the final places of the
##arcs r and r-1 in the new sequence.
        if sign[1]=sign[2] then
          if posn[2]=i+1 then
            smooth[posn[1]]:=r:     smooth[posn[2]]:=r-1:
            smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r:
          else
            smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r:
            smooth[posn[3]]:=r:     smooth[posn[4]]:=r-1:
          fi:
          mu2:=z:
        elif sign[1]<>sign[2] then
          if posn[2]=i+1 then
            smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r-1:
            smooth[posn[3]]:=r:     smooth[posn[4]]:=r:
          else
            smooth[posn[1]]:=r:     smooth[posn[2]]:=r:
            smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r-1:
          fi:
```

```
          mu2:=-z:
        fi:
##mu2 stores the value of the multiplier
##for the rearrangement, which in
##this program is always z or -z
      elif depth[1]=r then
        if sign[1]=sign[2] then
          if posn[1]=i+1 then
            smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r:
            smooth[posn[3]]:=r:     smooth[posn[4]]:=r-1:
          else
            smooth[posn[1]]:=r:     smooth[posn[2]]:=r-1:
            smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r:
          fi:
          mu2:=-z:
        elif sign[1]<>sign[2] then
          if posn[1]=i+1 then
            smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r-1:
            smooth[posn[3]]:=r:     smooth[posn[4]]:=r:
          else
            smooth[posn[1]]:=r:     smooth[posn[2]]:=r:
            smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r-1:
          fi:
          mu2:=z:
        fi:
      fi:
    fi:
##Now we have the routines that move
##the coefficients
####
##SeqIndex is a called program that finds
```

```
##the index of a required number sequence,
    y:=SeqIndex(A,switch):
    B[y]:=simplify(B[y]+B[j]):
    if mu2<>0 then
      y:=SeqIndex(A,smooth):
      B[y]:=simplify(B[y]+mu2*B[j]):
    fi:
    B[j]:=0:
  fi:             #End of routine for positive crossings
elif i<0 then     #Case for negative crossings
  if B[j]<>0 and A[j][abs(i)]=r
             and A[j][abs(i)+1]<r then
    depth:=[0,0,0,0]:    posn:=[0,0,0,0]:
    w:=1: mu2:=0:
    while w<5 do
      for c from 1 to 2*k do
        if A[j][c]=r-1 or A[j][c]=r then
          depth[w]:=A[j][c]: posn[w]:=c: w:=w+1:
        fi:
      od:
    od:
 ##In this section we have similar
 ##pieces of code to previously; these
 ##deal with the case when we need
 ##to ensure compatibility for an
 ##inverse
    switch:=A[j]:
    for f from 1 to 2*k do
      if A[j][f]=r-1 then
        switch[f]:=r:
      elif A[j][f]=r then
    switch[f]:=r-1:
  fi:
od:
if depth[1]=depth[3] then
  smooth:=A[j]:
  sign:=[S[posn[1]],S[posn[2]]]:
  if depth[1]=r-1 then
    if sign[1]=sign[2] then
      if posn[4]=abs(i) then
        smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r:
        smooth[posn[3]]:=r:     smooth[posn[4]]:=r-1:
      else
        smooth[posn[1]]:=r:     smooth[posn[2]]:=r-1:
        smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r:
      fi:
      mu2:=z:
    elif sign[1]<>sign[2] then
      if posn[4]=abs(i) then
        smooth[posn[1]]:=r:     smooth[posn[2]]:=r:
        smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r-1:
      else
        smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r-1:
        smooth[posn[3]]:=r:     smooth[posn[4]]:=r:
      fi:
      mu2:=-z:
    fi:
  elif depth[1]=r then
    if sign[1]=sign[2] then
      if posn[1]=abs(i) then
        smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r:
        smooth[posn[3]]:=r:     smooth[posn[4]]:=r-1:
            else
              smooth[posn[1]]:=r:     smooth[posn[2]]:=r-1:
              smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r:
            fi:
            mu2:=-z:
          elif sign[1]<>sign[2] then
            if posn[1]=abs(i) then
              smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r-1:
              smooth[posn[3]]:=r:     smooth[posn[4]]:=r:
            else
              smooth[posn[1]]:=r:     smooth[posn[2]]:=r:
              smooth[posn[3]]:=r-1:   smooth[posn[4]]:=r-1:
            fi:
            mu2:=z:
          fi:
        fi:
      fi:
      y:=SeqIndex(A,switch):
      B[y]:=simplify(B[y]+B[j]):
      if mu2<>0 then
        y:=SeqIndex(A,smooth):
        B[y]:=simplify(B[y]+mu2*B[j]):
      fi:
      B[j]:=0:
    fi:
  fi:         #End of routine for negative crossing
 od:          #End of loop through A,B
od:
########################################################
#END OF REARRANGEMENT/RENUMBERING LOOP
########################################################
```

```
#By this point in the algorithm, the array
#of coefficients has been rearranged so
#that the only k-sequences which can
#have non-zero coefficients are those
#which are compatible with the generator
#or inverse
###
#This is achieved after k-1 passes of the
#set of k-sequences
########################################################
#MULTIPLICATION PROCEDURE
########################################################
#This is a much shorter procedure, there
#is much less work to do in terms
#of searching through the arrays; we have
#two slightly different routines depending
#on whether or not we have a positive or
#negative crossing
########################################################
  if i>0 then
  ##Generator
    for t from 1 to m do
      if B[t]<>0 and (A[t][i]>=A[t][i+1]) then
        mult_temp:=A[t]:
        mult_temp[i]:=A[t][i+1]:
        mult_temp[i+1]:=A[t][i]:
        if mult_temp=A[t] then
          B[t]:=simplify(v*B[t]):
        else
          y:=SeqIndex(A,mult_temp):
          B[y]:=B[t]:
```

```
        B[t]:=0:
      fi:
    fi:
  od:
elif i<0 then
##Inverse
  for t from 1 to m do
    if B[t]<>0
    and (A[t][abs(i)]<=A[t][abs(i)+1]) then
      mult_temp:=A[t]:
      mult_temp[abs(i)]:=A[t][abs(i)+1]:
      mult_temp[abs(i)+1]:=A[t][abs(i)]:
      if mult_temp=A[t] then
        B[t]:=simplify(v^(-1)*B[t]):
      else
        y:=SeqIndex(A,mult_temp):
        B[y]:=B[t]:
        B[t]:=0:
      fi:
    fi:
  od:
fi:
##The following lines updates
##orientations of the the linear
##combination of k-sequences
S1:=S:
S1[abs(i)]:=S[abs(i)+1]:
S1[abs(i)+1]:=S[abs(i)]:
S:='S':
S:=S1:
S1:='S1':
```

```
########################################################
od:
########################################################
#END OF MAIN REARRANGEMENT AND MULTIPLICATION LOOP
########################################################
#CLOSURE ROUTINE
########################################################
#After all of the multiplications are complete
#we must close off each k-sequence because of
#the 'cups'
###
#We perform the closure one 'cup' at a time.
########################################################
#k is passed in, but it is only used to give a
#value to the first loop which controls the
#overall process and how many times
#it is repeated
########################################################
#INITIALISING CLOSURE PROCEDURE
########################################################
Anow:=A: Bnow:=B:
A:='A': B:='B': m:='m':
m:=ArrayNumElems(Anow):
delta:=((1/v)-v)/z:
##Define delta, value of disjoint unknot
########################################################
#START CLOSURE LOOP
########################################################
for f from k to 2 by -1 do
  if (nops(Y)/2)>2 then
  ##Don't need rearrangement
```

```
##for closure of 2-sequences
  for r from (nops(Y)/2) to 3 by -1 do
    for j from 1 to m do
      if (Bnow[j]<>0) and
         ((Anow[j][1]=r and Anow[j][2]<(r-1))
         or (Anow[j][2]=r and Anow[j][1]<(r-1))) then
        depth:=[0,0,0,0]: posn:=[0,0,0,0]:
        w:=1: mu2:=0:
        while w<5 do
          for c from 1 to nops(Y) do
            if Anow[j][c]=r or Anow[j][c]=r-1 then
              depth[w]:=Anow[j][c]: posn[w]:=c: w:=w+1:
            fi:
          od:
        od:
        switch:=Anow[j]:
        for i from 1 to nops(Y) do
          if Anow[j][i]=r then
            switch[i]:=r-1:
          elif Anow[j][i]=r-1 then
            switch[i]:=r:
          fi:
        od:
        ##Then we have the routine that will
        ##decide if we need rearrangement
        ##rather than renumbering
        if depth[1]=depth[3] then
          smooth:=Anow[j]:
          sign:=[S[posn[1]],S[posn[2]]]:
          if sign[1]<>sign[2] then
            smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r-1:
              smooth[posn[3]]:=r:     smooth[posn[4]]:=r:
              mu2:=z:
            elif sign[1]=sign[2] then
              smooth[posn[1]]:=r-1:   smooth[posn[2]]:=r:
              smooth[posn[3]]:=r:     smooth[posn[4]]:=r-1:
              mu2:=-z:
            fi:
          fi:
          ##Moving coefficients
          y:=SeqIndex(Anow,switch):
          Bnow[y]:=simplify(Bnow[y]+Bnow[j]):
          if mu2<>0 then
            y:=SeqIndex(Anow,smooth):
            Bnow[y]:=simplify(Bnow[y]+mu2*Bnow[j]):
          fi:
          Bnow[j]:=0:
        fi:
      od:
    od:
  fi:
##All sequences are closure-compatible
####
##Need to initialise variables that will be used
##for the next level of closure, i.e., to pass
##to sequences with one less arc
  Ynext:=Y[1..(nops(Y)-2)]:
  ##Ynext is the base generator string
  C:=permute(Ynext,nops(Ynext)):
  Anext:=Array(1..nops(C)):
  Bnext:=Array(1..nops(C)):
  ##Anext is the set of sequences that
```

```
##coefficients will be passed to
####
##Bnext stores corresponding
##coefficients for Anext
for a from 1 to nops(C) do
  Anext[a]:=C[a]: Bnext[a]:=0:
od:
C:='C':
##Perform closure
for j from 1 to m do
  if Bnow[j]<>0 then
    close_temp:=Anow[j][3..nops(Y)]:
    close_mult:=0:
    if Anow[j][1]=Anow[j][2] then
      close_mult:=delta*Bnow[j]:
    else
      depth:=[Anow[j][1],Anow[j][2],0,0]: w:=3:
      while w<5 do
        for b from 3 to nops(Y) do
          if (Anow[j][b]=Anow[j][1]) or (Anow[j][b]=Anow[j][2]) then
            depth[w]:=Anow[j][b]: w:=w+1:
          fi:
        od:
      od:
  ##The following determines the
  ##multiplier that is carried through
      if depth[2]=depth[3] then
        close_mult:=1*Bnow[j]:
      elif depth[2]<depth[3] then
        close_mult:=v*Bnow[j]:
      elif depth[2]>depth[3] then
```

```
          close_mult:=(1/v)*Bnow[j]:
        fi:
      fi:
    ##close_mult, by this point
    ##is the coefficient that is
    ##passed to the next stage
    ##accounting for any multiplier
      for i from 1 to nops(Ynext) do
        if close_temp[i]>min(Anow[j][1],Anow[j][2]) then
          close_temp[i]:=close_temp[i]-1:
        fi:
      od:
      y:=SeqIndex(Anext,close_temp):
      Bnext[y]:=simplify(Bnext[y]+close_mult):
    fi:
  od:
########################################################
#END CLOSURE
########################################################
##Have to initialise Anow, Bnow,
##Y and m for next loop.
####
##We also need to remove the
##first two elements of S.
  Anow:='Anow': Bnow:='Bnow': Y:='Y':
  Anow:=Anext: Bnow:=Bnext: Y:=Ynext:
  Anext:='Anext': Bnext:='Bnext': Ynext:='Ynext':
  m:=ArrayNumElems(Anow):
  S1:=S[3..nops(S)]:
  S:='S':  S:=S1:  S1:='S1':
########################################################
```

```
#END MAIN PROGRAM
########################################################
od:
########################################################
#FINAL OUTPUT STAGE
########################################################
output:=Bnow[1]:
##The polynomial of the plait presentation
##has been calculated, and we output this
##with coefficients of v on z
output:=collect(expand(output),z):
end;
```

Appendix B

Plait Presentations

In this chapter we give tables of representative words for plait presentations of knots up to ten crossings.

Unless otherwise stated the diagrams that the presentations are based on were taken from the Knot Atlas Rolfsen tables [52]. All presentations have minimal plait width, but not necessarily minimal crossing number for that width.

Knot	k	Presentation	Notes
3_1	2	2 -1 2	
4_1	2	-2 -2 3 -2	
5_1	2	2 -3 -3 -3 2	
5_2	2	-2 -2 1 3 -2	
6_1	2	-2 -2 1 1 3 -2	
6_2	2	2 -1 2 -1 -3 2	
6_3	2	-2 -2 3 -2 1 -2	
7_1	2	2 -1 -1 -3 -3 -3 2	
7_2	2	-2 -2 1 1 3 3 -2	
7_3	2	-2 -2 -2 1 1 3 -2	
7_4	2	2 -1 -3 2 -1 -3 2	
7_5	2	2 -1 2 2 -1 -3 2	
7_6	2	2 2 -3 -3 2 -3 2	
7_7	2	2 -3 2 -1 2 -3 2	

Table B.1: Plait presentations for knots of up to 7 crossings

Knot	k	Presentation	Notes
8_1	2	-2 -2 1 1 3 3 3 -2	
8_2	2	2 -1 2 -1 -3 -3 -3 2	
8_3	2	-2 -2 -2 -2 1 3 3 -2	
8_4	2	-2 -2 -2 1 -2 -2 -2 -2	
8_5	3	-2 -4 3 3 3 5 -2 -4	
8_6	2	2 -1 2 2 2 -1 -3 2	
8_7	2	-2 -2 3 -2 1 1 3 -2	
8_8	2	2 2 -1 -1 -1 2 -3 2	
8_9	2	-2 1 3 -2 1 -2 -2 -2	
8_{10}	3	2 1 -4 3 -2 1 3 5 -2 -4	10 crossing
8_{11}	2	2 -1 2 -3 -3 2 2 2	
8_{12}	2	-2 -2 1 1 -2 -2 3 -2	
8_{13}	2	2 -1 -3 2 -3 2 -1 2	
8_{14}	2	-2 -2 3 -2 3 3 -2 -2	
8_{15}	3	2 4 -5 4 -3 4 4 2	
8_{16}	3	2 4 -3 -5 4 -3 2 4	
8_{17}	3	2 4 -3 2 -5 4 -3 2 5 4	10 crossing
8_{18}	3	-2 -4 3 -2 -4 3 -2 -4	
8_{19}	3	-2 -4 1 -3 2 -3 -4 -2	
8_{20}	3	2 4 3 2 -4 -3 -5 4 1 2	10 crossing
8_{21}	3	-2 4 3 4 -2 3 -5 4 1 2	10 crossing

Table B.2: Plait presentations for knots with 8 crossings

Knot	k	Presentation	Notes
9_1	2	2 -1 -1 -1 -3 -3 -3 -3 2	
9_2	2	-2 -2 1 1 1 3 3 3 -2	
9_3	2	-2 -2 -2 1 1 1 1 3 -2	
9_4	2	-2 -2 -2 -2 1 1 3 3 -2	
9_5	2	-2 -2 -2 1 -2 -2 -2 -2 -2	
9_6	2	2 -1 -3 -3 -3 2 2 -1 2	
9_7	2	2 -1 2 2 2 2 -1 -3 2	
9_8	2	2 2 -3 -3 -3 -3 2 -1 2	
9_9	2	2 -1 -1 -3 2 2 -1 -3 2	
9_{10}	2	2 -1 -3 2 2 2 -1 -3 2	
9_{11}	2	-2 -2 3 3 -2 1 1 3 -2	
9_{12}	2	2 -3 2 -1 -1 2 2 2 2	
9_{13}	2	2 -3 -3 2 -1 -1 2 2 2	
9_{14}	2	2 -1 -1 -3 2 -1 2 -3 2	
9_{15}	2	-2 -2 1 3 -2 -2 -2 1 -2	
9_{16}	3	2 2 2 4 5 -1 -3 2 2 2 4	11 crossing
9_{17}	2	-2 -2 1 -2 -2 -2 1 -2 -2	
9_{18}	2	2 2 2 -1 -1 2 2 -3 2	
9_{19}	2	-2 -2 3 -2 3 3 3 -2 -2	
9_{20}	2	-2 -2 3 -2 -2 3 -2 -2 -2	
9_{21}	2	-2 -2 -2 3 -2 3 3 -2 -2	
9_{22}	3	-2 1 -2 -4 3 3 3 -2 -4	
9_{23}	2	-2 -2 1 1 -2 1 1 -2 -2	
9_{24}	3	2 2 -3 4 5 -3 -3 2 2 2 4	11 crossing
9_{25}	3	2 2 4 -3 2 2 -1 2 4	

Table B.3: Plait presentations for knots 9_1 to 9_{25}

Knot	k	Presentation	Notes
9_{26}	2	-2 1 3 -2 1 -2 1 -2 -2	
9_{27}	2	2 -3 2 -1 2 2 -1 2 2	
9_{28}	3	2 -3 2 4 5 -1 -3 2 -1 2 4	11 crossing
9_{29}	3	-2 4 5 3 -2 -4 3 -2 -4 3 -4 1 2	13 crossing
9_{30}	3	2 2 4 -3 2 -1 2 2 4	
9_{31}	2	2 -3 2 -1 2 -1 2 -3 2	
9_{32}	3	-2 -4 3 -2 3 3 -2 -4 -4	
9_{33}	3	2 1 4 -3 2 4 -1 -3 2 2 4	11 crossing
9_{34}	3	-2 -4 3 -2 -4 3 3 -2 -4	
9_{35}	3	-2 -4 1 3 3 3 5 -2 -4	
9_{36}	3	-2 -4 -4 1 3 3 5 -2 -4	
9_{37}	3	2 4 -5 4 -3 2 4 4 4	
9_{38}	3	2 1 4 -3 2 -5 4 -3 2 4 4	11 crossing
9_{39}	3	2 4 -3 -5 4 -3 2 4 4	
9_{40}	3	2 4 -3 4 -3 2 -3 2 4	
9_{41}	3	-2 -4 3 3 5 -4 3 -2 -4	
9_{42}	3	-2 -4 -4 1 -3 -3 5 -2 -4	
9_{43}	3	-2 4 4 3 -5 4 3 -2 4	
9_{44}	3	-2 4 3 -2 -3 2 -1 2 4	
9_{45}	3	-2 -4 1 -2 3 3 -5 2 4	
9_{46}	3	-2 -4 1 3 3 3 -5 -2 4	
9_{47}	3	-2 -4 -3 -3 -2 -4 3 -2 -4	
9_{48}	3	2 4 -1 -3 -2 3 -2 -2 4	
9_{49}	3	2 4 -3 -3 -5 -4 3 2 -4	

Table B.4: Plait presentations for knots 9_{26} to 9_{49}

Knot	k	Presentation	Notes
10_1	2	-2 -2 1 1 1 1 1 1 3 -2	
10_2	2	2 -1 -1 -1 -1 -1 -1 2 -1 2	Knot redrawn
10_3	2	-2 -2 -2 -2 1 1 1 1 3 -2	
10_4	2	2 -1 -1 -1 -1 -3 2 -1 -1 2	Knot redrawn
10_5	2	-2 -2 3 -2 1 3 3 3 3 -2	
10_6	2	2 -1 2 2 2 -1 -3 -3 -3 2	
10_7	2	-2 -2 3 -2 -2 1 1 1 3 -2	
10_8	2	-2 -2 -2 -2 1 -2 -2 -2 -2 -2	
10_9	2	-2 1 1 1 3 -2 1 -2 -2 -2	
10_{10}	2	2 -1 -1 -1 -3 2 -3 2 -1 2	
10_{11}	2	2 -1 -1 -3 2 2 2 -1 -3 2	
10_{12}	2	-2 1 1 3 -2 -2 -2 1 -2 -2	
10_{13}	2	-2 1 3 3 -2 -2 1 1 -2 -2	
10_{14}	2	2 -1 -1 -3 2 2 -3 2 -1 2	
10_{15}	2	-2 1 1 3 -2 1 1 1 -2 -2	
10_{16}	2	-2 1 3 3 -2 1 1 -2 -2 -2	
10_{17}	2	-2 1 3 3 -2 1 -2 -2 -2 -2	
10_{18}	2	2 -1 -1 -3 2 -3 2 2 -1 2	
10_{19}	2	2 -1 -1 -3 2 -3 2 -1 -3 2	
10_{20}	2	-2 -2 -2 3 3 3 3 3 -2 -2	
10_{21}	2	2 2 2 -3 -3 -3 -3 2 -1 2	
10_{22}	2	-2 1 3 -2 -2 -2 1 -2 -2 -2	
10_{23}	2	-2 -2 3 -2 3 3 3 -2 -2 -2	
10_{24}	2	-2 1 3 -2 -2 1 1 1 -2 -2	
10_{25}	2	2 -1 -3 2 2 -3 -3 2 -1 2	

Table B.5: Plait presentations for knots 10_1 to 10_{25}

Knot	k	Presentation	Notes
10_{26}	2	2 -1 -3 2 -1 2 2 -1 -3 2	
10_{27}	2	-2 -2 -2 1 1 -2 3 3 -2 -2	Knot redrawn
10_{28}	2	2 -3 2 -1 -1 -1 2 -1 -3 2	
10_{29}	2	2 -1 2 2 -1 -1 2 -1 -3 2	
10_{30}	2	-2 1 3 -2 1 1 -2 1 -2 -2	
10_{31}	2	-2 -2 -2 3 -2 3 3 3 -2 -2	
10_{32}	2	2 2 2 -3 2 -3 2 2 -1 2	Knot redrawn
10_{33}	2	-2 1 3 -2 -2 -3 -3 2 -1 -3 2	11 crossing; knot redrawn
10_{34}	2	2 -3 2 -1 -1 -1 -1 -1 2 2	
10_{35}	2	2 -1 2 2 -1 -1 -1 -1 2 2	
10_{36}	2	-2 -2 1 1 1 1 -2 1 -2 -2	
10_{37}	2	2 -3 2 2 2 -1 -1 -1 2 2	
10_{38}	2	2 2 -1 -3 2 -3 -3 -3 2 2	KnotInfo diagram
10_{39}	2	-2 -2 1 -2 -2 -2 3 3 -2 -2	
10_{40}	2	2 -3 2 2 -3 -3 2 -3 2 2	KnotInfo diagram
10_{41}	2	2 -1 2 -1 -1 2 -1 -1 2 2	
10_{42}	2	-2 1 -2 3 -2 -2 -3 -3 -3 2 2	11 crossing; KnotInfo diagram
10_{43}	2	2 2 -1 2 2 -3 -3 2 -1 2	
10_{44}	2	2 -1 2 -1 2 -1 -1 2 -1 2	
10_{45}	2	2 -1 2 -1 2 -1 2 -1 2 2	
10_{46}	3	-2 -4 3 3 3 5 5 5 -2 -4	
10_{47}	3	2 1 -4 3 -2 1 3 5 5 5 -2 -4	12 crossing
10_{48}	3	4 5 -2 -2 -2 -2 3 3 3 5 -2 -4	12 crossing
10_{49}	3	4 5 -2 -2 -2 -2 3 -4 3 5 -2 -4	12 crossing
10_{50}	3	-2 -4 3 3 3 5 -2 -4 -4 -4	

Table B.6: Plait presentations for knots 10_{26} to 10_{50}

Knot	k	Presentation	Notes
10_{51}	3	-2 -4 1 3 5 -2 3 -4 -4 -4 1 2	12 crossing
10_{52}	3	-2 -4 3 3 3 -4 5 5 -2 -4	
10_{53}	3	-2 -4 1 3 -2 3 -4 5 5 -4 3 2	12 crossing
10_{54}	3	-2 -4 3 3 3 5 5 -2 -4 -4	
10_{55}	3	-2 -4 1 3 5 5 -2 3 -4 -4 1 2	12 crossing
10_{56}	3	-2 -4 3 3 3 -4 -4 5 -2 -4	
10_{57}	3	-2 -4 1 3 -2 3 -4 -4 5 -4 1 2	12 crossing
10_{58}	3	-2 -4 1 3 3 -2 3 5 -4 -4 1 2	12 crossing
10_{59}	3	2 1 -4 -4 3 3 5 -2 1 3 -2 -4	12 crossing
10_{60}	3	2 2 -3 4 2 -3 2 2 4 4	KnotInfo diagram
10_{61}	3	-2 -4 1 3 3 3 5 5 -2 -4	
10_{62}	3	-2 -4 3 3 3 5 5 -2 -2 -4	
10_{63}	3	2 -4 -4 3 5 -4 3 5 5 -2 3 4	12 crossing
10_{64}	3	-2 -2 -2 -4 3 3 3 5 -2 -4	
10_{65}	3	-2 -2 -4 3 3 3 -2 -4 -4 -4	
10_{66}	3	2 -4 -4 3 5 -4 3 5 -4 -4 -4 -2	12 crossing
10_{67}	3	-2 -4 3 3 3 5 -2 -2 -4 -4	
10_{68}	3	-2 -4 1 3 3 3 -4 5 -2 -4	
10_{69}	3	2 -4 -4 3 5 -4 3 5 -4 -2 3 -4	12 crossing
10_{70}	3	2 -1 4 5 3 4 -5 -2 -2 -2	19 crossing; KnotInfo diagram
		-1 4 -5 2 3 1 2 4 4	
10_{71}	3	-2 -2 -4 1 3 -2 -2 1 -2 -4	KnotInfo diagram
10_{72}	3	2 2 -3 4 -3 2 2 2 4 4	
10_{73}	3	2 2 4 4 -1 -1 -3 -2 -2 -2 -4	11 crossing; KnotInfo diagram
10_{74}	3	-2 3 4 5 -2 1 3 -2 -2 -2 -4 -4	12 crossing; KnotInfo diagram
10_{75}	3	-2 3 -4 -2 1 3 -2 1 -2 -4	KnotInfo diagram

Table B.7: Plait presentations for knots 10_{51} to 10_{75}

Knot	k	Presentation	Notes
10_{76}	3	2 2 2 4 -1 -5 -4 -3 2 2 2 -4	12 crossing; KnotInfo diagram
10_{77}	3	-2 -2 -4 1 1 3 -2 -2 -2 -4	KnotInfo diagram
10_{78}	3	2 -1 2 -1 -1 4 5 -3 2 -1 2 4	12 crossing
10_{79}	3	-2 -4 3 5 -4 -4 -4 2 3 2 4 -3	12 crossing
10_{80}	3	-2 -2 -2 4 5 3 -4 3 -2 -2 5 -4	12 crossing
10_{81}	3	-2 -4 -4 1 3 -2 3 -4 5 -4 1 2	12 crossing
10_{82}	3	2 4 -3 -5 -5 2 -1 2 -3 4 1 2	12 crossing
10_{83}	3	2 -1 -3 4 2 -3 2 -1 2 -3 -4 2	12 crossing
10_{84}	3	2 4 -3 4 -5 4 -3 -5 -5 -2 4 4	12 crossing
10_{85}	3	2 4 -3 2 -3 -3 -5 -5 2 4	
10_{86}	3	2 4 -3 4 -3 -3 4 1 2 -3 -5 4	12 crossing
10_{87}	3	2 4 4 -3 2 -3 -3 -5 2 4	
10_{88}	3	2 2 4 5 3 4 -5 4 -3 2 4 -3 2 4	14 crossing
10_{89}	3	-2 4 5 3 -2 3 -4 -2 1 -2 5 -4	12 crossing
10_{90}	3	-2 4 5 3 -2 -4 3 5 -2 -2 -2 -4	12 crossing
10_{91}	3	2 1 4 -3 2 4 -1 -5 -5 2 -3 1 2 4	14 crossing
10_{92}	3	2 4 4 -3 -5 2 4 4 -3 4 1 2	12 crossing
10_{93}	3	2 1 4 -3 -3 4 -5 -5 -3 -3 2 4	12 crossing
10_{94}	3	4 5 2 2 2 -3 2 4 -3 -3 2 4	12 crossing
10_{95}	3	2 -3 4 5 2 -3 2 4 -3 -3 2 4	12 crossing
10_{96}	3	2 -3 -4 -5 -3 -3 2 4 -3 -3 2 4	12 crossing
10_{97}	3	2 2 4 -3 -5 4 -3 -3 2 4	KnotInfo diagram
10_{98}	3	2 4 -3 -3 2 2 -1 -3 2 4	
10_{99}	3	2 1 4 -3 2 -1 -5 4 2 -3 -5 1 2 4	14 crossing
10_{100}	3	2 4 -1 -3 -5 2 -1 -3 2 4	

Table B.8: Plait presentations for knots 10_{76} to 10_{100}

Knot	k	Presentation	Notes
10_{101}	3	2 -4 -4 3 -4 5 -4 3 -2 -4 3 -4	12 crossing
10_{102}	3	-2 -4 1 3 -2 -4 3 -4 -4 -4 1 2	12 crossing
10_{103}	3	2 1 -4 -4 3 -2 -4 1 5 -4 5 4 3 -2	14 crossing
10_{104}	3	2 -3 4 5 -4 -4 1 5 -2 3 -2 -4	12 crossing
10_{105}	3	2 1 -4 3 3 5 -4 3 -2 1 -2 -4	12 crossing
10_{106}	3	2 3 -4 3 -2 1 1 5 -4 3 -2 -4	12 crossing
10_{107}	3	-2 -2 4 5 3 -4 5 5 -4 3 -2 -4	12 crossing
10_{108}	3	-2 4 3 -2 1 3 3 5 -2 -4	
10_{109}	3	-2 -4 3 3 -2 -2 3 3 -2 -2 5 4	12 crossing
10_{110}	3	-2 4 5 3 -2 -2 -4 3 3 -2 -2 -4	12 crossing
10_{111}	3	2 1 4 4 -3 2 4 4 -5 -5 -4 -3 -4 2	14 crossing; knot redrawn
10_{112}	3	2 1 -4 3 -4 3 -2 3 -2 -2 -2 -4	12 crossing
10_{113}	3	2 4 -3 2 -1 2 -3 2 4 4	
10_{114}	3	-2 -4 3 3 3 -2 -4 3 -2 -4	
10_{115}	3	2 1 -4 3 5 -2 -4 3 -4 3 -2 -4	12 crossing
10_{116}	3	2 4 -3 -5 2 4 -3 -5 2 4	
10_{117}	3	-2 -4 3 -2 3 -4 3 3 1 2 -4 -4	12 crossing
10_{118}	3	-2 4 5 3 -2 3 3 5 -4 3 1 2 -4 -4	14 crossing
10_{119}	3	2 -4 3 4 -1 2 -5 4 -3 -3 2 4	12 crossing; knot redrawn
10_{120}	3	-2 -4 3 3 -2 -4 3 3 -2 -4	
10_{121}	3	-2 4 5 3 -2 3 -4 -2 3 5 -2 -4	12 crossing
10_{122}	3	2 4 -3 4 -3 -3 2 -3 2 4	
10_{123}	3	2 4 -3 2 4 -5 4 -3 2 4	
10_{124}	3	2 -4 3 2 3 3 2 5 5 -4	
10_{125}	3	-2 -4 -3 -3 -3 5 5 5 -2 -4	

Table B.9: Plait presentations for knots 10_{101} to 10_{125}

Knot	k	Presentation	Notes
10_{126}	3	-2 4 3 3 3 -5 -5 -5 -2 4	
10_{127}	3	2 -4 -1 3 -2 3 1 2 -4 -4 -4 -4	12 crossing
10_{128}	3	-2 -4 -1 -3 2 -3 -2 -2 -2 -4	
10_{129}	3	-2 4 3 -5 -4 3 -4 -4 -4 -2	
10_{130}	3	2 -4 3 3 3 4 2 3 5 -4	
10_{131}	3	-2 -4 1 -3 -3 -3 -4 1 2 3 5 -4	12 crossing
10_{132}	3	-2 -4 -3 2 4 -3 5 -2 -4 -4	
10_{133}	3	2 -4 3 2 -1 3 2 3 -4 -4	
10_{134}	3	-2 4 1 3 -5 -2 -3 2 4 4	
10_{135}	3	2 -4 -1 3 -2 3 -4 -4 5 1 2 -4	12 crossing
10_{136}	3	-2 4 3 -2 1 -2 3 -2 4 4	
10_{137}	3	2 2 2 1 3 -4 5 -3 2 4 3 -2 -2 5 4	15 crossing; knot redrawn
10_{138}	3	-2 1 1 -4 -4 2 -3 4 1 2 -3 4 4	13 crossing; KnotInfo diagram
10_{139}	3	2 1 -4 -3 5 2 2 4 1 -3 -2 -4	12 crossing
10_{140}	3	2 -4 -1 3 3 3 5 5 2 -4	
10_{141}	3	2 4 -3 -5 -4 5 2 3 2 -4	
10_{142}	3	2 -4 -1 -1 3 3 3 5 2 -4	
10_{143}	3	2 1 -4 3 -2 1 5 -4 -5 2 3 4 1 2	14 crossing
10_{144}	3	-2 -2 -4 -3 -3 -3 -3 -2 -4 -4	
10_{145}	3	2 1 4 3 3 4 4 3 -2 3 2 4	12 crossing
10_{146}	3	4 -3 2 3 -5 -4 3 5 5 -4 -4 -2	12 crossing
10_{147}	3	2 4 -1 -3 -2 -1 3 -2 4 4	
10_{148}	3	-2 4 3 -5 -2 -4 1 -4 3 -2 5 4	12 crossing
10_{149}	3	-2 4 5 1 3 -4 -4 -2 3 1 2 -3 -5 4	14 crossing
10_{150}	3	-2 4 3 -2 4 3 5 -2 4 4	

Table B.10: Plait presentations for knots 10_{126} to 10_{150}

Knot	k	Presentation	Notes
10_{151}	3	2 -1 -4 -4 3 -2 3 -4 5 -4 1 2	12 crossing
10_{152}	3	2 4 5 3 -1 -2 -4 -4 3 -5 -5 4 1 2	14 crossing
10_{153}	3	2 -4 -4 -1 -3 2 -3 -4 -4 -4 1 2	12 crossing
10_{154}	3	2 -4 -4 -1 -3 2 -3 -4 5 -4 1 2	12 crossing
10_{155}	3	-2 4 1 -3 -5 4 -3 -5 -2 4	
10_{156}	3	2 1 -4 -3 -4 -5 -4 2 -1 -3 2 2 5 4	14 crossing
10_{157}	3	2 1 -4 5 -3 2 -1 -4 2 -3 5 -4 1 2	14 crossing
10_{158}	3	2 4 -1 -3 -3 -2 3 -2 -2 4	
10_{159}	3	2 4 -3 2 -4 -4 -4 -3 2 4	
10_{160}	3	-2 -4 3 -4 -3 -3 2 -3 -2 -4	
10_{161}	3	4 5 2 3 -1 -2 -4 -4 3 5 4 -2	12 crossing
10_{162}	3	-2 4 3 -2 -1 -3 -3 5 2 4	
10_{163}	3	-2 -4 3 -2 -4 -3 -3 -3 -2 -4	
10_{164}	3	2 -4 3 -4 -3 -3 2 -3 2 4	
10_{165}	3	2 4 -3 -3 2 4 3 3 2 4	

Table B.11: Plait presentations for knots 10_{151} to 10_{165}

Bibliography

[1] J. W. Alexander "A lemma on a system of knotted curves" *Proc. Nat. Acad. Sci. USA.* **9** (1923) 93 – 95

[2] J. W. Alexander "Topological invariants of knots and links" *Trans. Amer. Math. Soc.* **30** (1928) 275 – 306

[3] E. Artin "Theorie der Zöpfe" *Abhandlungen Math. Seminar Univ. Hamburg* **4** (1925) 47 – 72

[4] E. Artin "Theory of braids" *Annals of Math.* **48** (1947) 101 – 126

[5] A. Beliakova & C. Blanchet "Skein construction of idempotents in Birma-Murakami-Wenzl algebras" *Mathematische Annalen* **321 (2)** (2001) 346 – 374

[6] J. S. Birman "Braids, links and mapping class groups" *Annals of Math. Studies* **82** Princeton Univ. Press. N.J. (1976)

[7] J. S. Birman & T. Kanenobu "Jones' braid-plat formula and a new surgery triple" *Proc. AMS* **102** No. 3 (1988) 687 – 695

[8] J. S. Birman, K. H. Ko & S. J. Lee "A new approach to the word and conjugacy problems in the braid groups" *Advances Math.* **139** (1998) 322 – 353

[9] G. Burde & H. Zieschang "Knots" (2nd Edition) *de Gruyter Studies in Mathematics* (2003)

[10] W. Y. C. Chen, E. Y. P. Deng, R. R. X. Du, R. P. Stanley, C. H. Yan "Crossings and Nestings of Matchings and Partitions" *Retrieved from the arXiv at* `http://arxiv.org/abs/math/0501230`

[11] S. V. Chmutov, S. V. Duzhin & S. K. Lando "Vassiliev knot invariants" *Adv. Soviet Math.* **21** (1994) 117 – 126

[12] J. H. Conway "An enumeration of knots and links, and some of their algebraic properties" *Computational Problems in Abstract Algebra (Proc. Conference Oxford 1970)*, ed J. Leech, Pergamon Press (1967) 320 – 358

[13] D. Cooper & W. B. R. Lickorish "Mutations of links in genus 2 handlebodies" *Proc. Amer. Math. Soc.* **127** (1999) 309 – 314

[14] P. R. Cromwell "Knots and Links" *Cambridge University Press* (2004)

[15] N. M. Dunfield, S. Garoufalidis, A. Shumakovitch & M. Thistlethwaite "Behaviour of knot invariants under genus 2 mutation" *Preprint* `http://arxiv.org/abs/math/0607258v1`

[16] J. Franks & R. F. Williams "Braids and the Jones Polynomial" *Trans. Amer. Math. Soc.* **303** (1987) 97 – 108

[17] P. Freyd, D. Yetter, J. Hoste, W. B. R. Lickorish, K. Millett, A. Ocneau "A new polynomial invariant of knots and links" *Bull. Amer. Math. Soc.* **12** (1985) 239 – 246

[18] R. J. Hadji "Homfly Skein Theory of Reversed String Satellites" *PhD Thesis* (2003)

[19] R. J. Hadji & H. R. Morton "A basis for the full Homfly skein of the annulus." *Math. Proc. Camb. Philos. Soc.* **141** (2006), 81 – 100

[20] N. Imafuji & M. Ochiai, Knot theory software.

`http://amadeus.ics.nara-wu.ac.jp/~ochiai/freesoft.html`

[21] I. M. Isaacs “Algebra: A Graduate Course.” Brooks/Cole Publishing Company, California (1994)

[22] V. F. R. Jones “A polynomial invariant for knots via von Neumann algebras” *Bull. American Math. Soc.* (1985) **12** 103 - 111

[23] E. S. Kang, K. H. Ko & S. J. Lee “Band-generator presentation for the 4-braid group” *Topology and its Applications* **78** (1997) 39 – 60

[24] L. H. Kauffman “On knots” *Princeton University Press* (1987)

[25] L. H. Kauffman “An invariant of regular isotopy” *Trans. Amer. Math. Soc.* (1990) **318** 417 – 471

[26] KnotInfo `http://www.indiana.edu/~knotinfo/`

[27] R. A. Landvoy “The Jones polynomial of pretzel knots and links” *Topology and its Applications* **83** (1998) 135 – 147

[28] W. B. R. Lickorish “Polynomials for links” *Bull. London. Math. Soc.* (1988) **20** 558 – 588

[29] W. B. R. Lickorish “An Introduction to Knot Theory” *Springer* (1997)

[30] W. B. R. Lickorish & A. S. Lipson “Polynomials of 2-cable-like links” *Proc. Amer. Math. Soc.* **100** (1987) 355 – 361

[31] B. Lu & J. K. Zhong “The Kauffman Polynomials of Generalized Hopf Links” *J. Knot Theory Ramif.* **11** (2002) 1291 – 1306

[32] B. Lu & J. K. Zhong “The Kauffman Polynomials of Pretzel Knots” *J. Knot Theory Ramif.* **17 (2)** (2008) 157 – 169

[33] I. G. Macdonald “Symmetric functions and Hall polynomials” Clarendon Press, Oxford, 2nd edition (1995)

[34] MorseLink Presentations: *retrieved from*

`http://katlas.math.toronto.edu/wiki/MorseLink_Presentations`

[35] H. R. Morton "Seifert Circles and Knot Polynomials" *Math. Proc. Cambridge Philos. Soc.* **99** (1986) 107 – 109

[36] H. R. Morton "Threading knot diagrams" *Math. Proc. Camb. Phil. Soc.* **99** (1986) 247 – 260

[37] H. R. Morton. "Quantum invariants given by evaluation of knot polynomials" *J. Knot Theory Ramifications* **2** (1993) 195 – 209

[38] H. R. Morton "Skein theory and the Murphy operators" *J. Knot Theory Ramifications* **11** (2002) 475 – 492

[39] H. R. Morton "Integrality of Homfly 1-tangle invariants." To appear in *Algebraic & Geometric Topology*

[40] H. R. Morton "Mutant knots with symmetry" *Retrieved from the arXiv at* `http://arxiv.org/abs/0705.1321`

[41] H. R. Morton & E. Beltrami "Arc index and the Kauffman polynomial." *Math. Proc. Camb. Philos. Soc.* **123** (1998) 41 – 48

[42] H. R. Morton & P. R. Cromwell "Distinguishing mutants by knot polynomials" *J. Knot Theory Ramifications* **5** (1996) 225 – 238

[43] H. R. Morton & H. J. Ryder "Mutants and $SU(3)_q$ invariants" in "Geometry and Topology Monographs", Vol.1: The Epstein Birthday Schrift (1998) 365 – 381

[44] H. R. Morton & N. Ryder "Invariants of genus 2 mutants" *Preprint* `http://arxiv.org/abs/0708.0514v1`

[45] H. R. Morton & H. B. Short "Calculating the 2-variable polynomial for knots presented as closed braids" *J. Algorithms* **11** (1990) 117 – 131

[46] H. R. Morton & P. Traczyk "The Jones polynomial of satellite links around mutants" In 'Braids', ed. Joan S. Birman and Anatoly Libgober, Contemporary Mathematics 78, *Amer. Math. Soc.* (1988) 587 – 592

[47] J. Murakami "Finite type invariants detecting the mutant knots" in "Knot Theory," a volume dedicated to Professor Kunio Murasugi for his 70th birthday. Ed. M. Sakuma et al., Osaka University (2000) 258 – 267

[48] J. H. Przytycki "Equivalence of cables of mutants of knots" *Canad. J. Math.* **41 (2)** (1989) 250 – 273

[49] J. H. Przytycki "Polynomial time complexity algorithm for computing coefficients of the Jones-Conway (Homflypt) and Kauffman polynomials of links" *Abstracts AMS* **23(1)** (2002) 147

[50] J. H. Przytycki & P. Traczyk "Invariants of links of Conway type" *Kobe J. Math.* **4** (1987) 115 – 139

[51] K. Reidemeister "Knotentheorie" *Ergebn. Math.* **1** (1932)

[52] Rolfsen Knot Table on Knot Atlas

`http://katlas.math.toronto.edu/wiki/The_Rolfsen_Knot_Table`

[53] D. Ruberman "Mutation and volume of knots in S^3" *Invent. Math.* **90** (1987) 189 – 215

[54] L. Rudolph "A congruence between link polynomials." *Math. Proc. Camb. Philos. Soc.* **107** (1990) 319 – 327

[55] H. Schubert "Knoten und Vollringe" *Acta Math.* **90** (1953) 131 – 286

[56] H. Schubert "Uber eine numerische Knoteninvariante" *Math. Z.* **61** (1954) 245 – 288

[57] A. Stoimenow & T. Tanaka "Mutation and the colored Jones polynomial" *Preprint* `http://arxiv.org/abs/math/0607794v2`

[58] H. F. Trotter "Non-invertible Knots Exist" *Topology* **2** (1964), 275 – 280

[59] V. G. Turaev "Operator Invariants of Tangles, and R-Matrices" *Math USSR Izv* **35 (2)** (1990) 411 – 444

[60] P. Vogel "Representation of links by braids: A new algorithm" *Comment. Math. Helvetici* **65** (1990) 104 – 113

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