Knot Invariants

Ali Ramsey

1 Introduction

The subject of this report is *knots* (and, more generally, *links*). Intuitively, if one were to take a piece of string, tie a knot in it, and fuse the ends together, the result would be a (mathematical) knot. A union of several of such knots (possibly linked together) would be an example of a link. We give a formal definition of the two below.

Definition 1.0.1. A *link* with m components is the image of a smooth embedding in $\mathbb{R}^3$ of m disjoint circles. A link with one component is a *knot*.

Definition 1.0.2. Two links L_1 and L_2 are equivalent if there is an orientation-preserving piecewise linear homeomorphism $h : \mathbb{R}^3 \to \mathbb{R}^3$ such that $h(L_1) = L_2$.

Our main focus will be *invariants* - mathematical entities (such as numbers and polynomials) associated with links, such that any two equivalent links will have equal invariants. In this way, given two link diagrams D_1 and D_2, we may prove they do *not* represent the same underlying link by proving that an invariant calculated from D_1 is not equal to the same invariant calculated from D_2.

Two polynomial invariants, the Jones polynomial and the Alexander polynomial, will be explored in particular detail in sections 2-5. In this section, we give basic definitions needed for the subsequent sections, as well as a statement of Reidemeister's theorem, which will underlie much of the later discussion of invariants.

Definition 1.0.3. A *framed link* is a link regarded as having some 'thickness'; that is, it consists of disjoint annuli embedded in $\mathbb{R}^3$ (rather than circles, as in Definition 1.0.1).

Definition 1.0.4. The *unknotting number* $u(L)$ of a link L is the minimum number of crossing changes needed to change L to the trivial link. It can be shown (by induction on the number of crossings in a diagram of L) that this is defined for every link.

Theorem 1.0.5 (Reidemeister). Let L_1, L_2 be two links and D_1, D_2 diagrams of them. Then, L_1 and L_2 are isotopic in $\mathbb{R}^3$ if and only if D_1 and D_2 are related by a sequence of isotopies in $\mathbb{R}^2$ and the RI, RII, RIII moves shown in Figure 1.1.

Finally, we will occasionally refer to knots using notation such as 3_1, denoting the first knot with 3 crossings, or to links as, for example, L6a2 (the second alternating link with 6 crossings) or L7n1 (the first non-alternating link with 7 crossings). All numbering (and notational) conventions are consistent with [1], where details about specifically mentioned knots and links can be found.

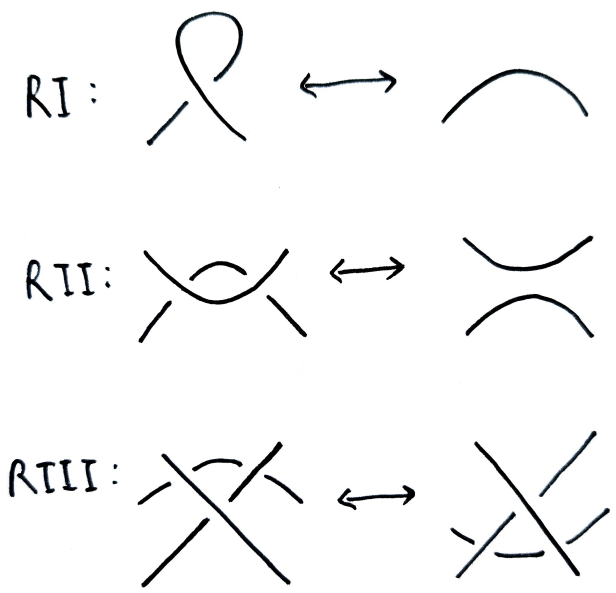


Figure 1.1: The three 'Reidemeister moves' RI, RII and RIII.

2 Invariants from skein relations

In this section, we will define the two focal invariants in terms of the skein relations they satisfy. Though this is not historically the way in which they were initially developed, it is the simplest place to start and we will refer back to these skein relations often in later sections.

Definition 2.0.1. A *skein relation* is a relationship between three link diagrams L_+, L_-, L_0 which differ only in the neighbourhood of a single point, where they appear as shown in Figure 2.1.

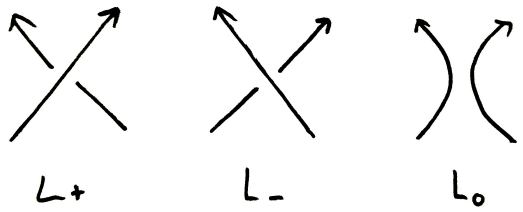


Figure 2.1: The links L_+, L_- and L_0 in the neighbourhood in which they differ.

2.1 The Jones polynomial

To define the Jones polynomial, we must first introduce an auxiliary function, known as the Kauffman bracket polynomial. We mostly follow [3] (chapter 3).

Definition 2.1.1. The *Kauffman bracket* of a link diagram D is a Laurent polynomial in an indeterminate A, characterised by

1. $\left\langle \bigcirc \right\rangle = 1$,
2. $\left\langle D \sqcup \bigcirc \right\rangle = (-A^2 - A^{-2})\langle D \rangle$,

3. $\left\langle \times \right\rangle = A \left\langle) (\right\rangle + A^{-1} \left\langle) (\right\rangle$.

From this definition we see that given any link diagram D with n crossings, the bracket $\langle D \rangle$ can be written as the linear sum of the brackets of 2^n diagrams with no crossings.

THEOREM 2.1.2. The Kauffman bracket of a link diagram is invariant under the Reidemeister moves RII and RIII.

PROOF. We first consider RII:

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

We now consider RIII:

$$\begin{aligned}
\left\langle \right\rangle &= A^{-1} \left\langle \right\rangle + A \left\langle \right\rangle \\
&= A^{-1} \left\langle \right\rangle + A \left\langle \right\rangle \quad \text{(using the previous part)} \\
&= A^{-1} \left\langle \right\rangle + A \left\langle \right\rangle \\
&= \left\langle \right\rangle .
\end{aligned}$$

□

LEMMA 2.1.3. We have the following:

$$\left\langle \right\rangle = -A^3 \left\langle \right\rangle ,$$

$$\left\langle \right\rangle = -A^{-3} \left\langle \right\rangle .$$

PROOF.

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

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

□

Definition 2.1.4. The *writhe* $w(D)$ of a diagram D of an oriented link is the sum of the signs of the crossings of D, as defined in Figure 2.1. Note that the writhe is invariant under RII and RIII.

Theorem 2.1.5. Let D be a diagram of an oriented link L. Then the expression

$$(-A)^{-3w(D)}\langle D\rangle$$

is an invariant of L.

Proof. By Theorem 2.1.2, $\langle D\rangle$ is invariant under the Reidemeister moves RII and RIII. Since the writhe also doesn't change under these moves, $V(L)$ is invariant under RII and RIII. From Lemma 2.1.3, we can see that $V(L)$ is also invariant under RI. Thus, by Reidemeister's theorem, $V(L)$ is an invariant of L.

□

Definition 2.1.6. The Jones polynomial $V(L)$ of an oriented link L is defined by

$$V(L) = \left((-A)^{-3w(D)}\langle D\rangle\right)_{A^{-2}=t^{1/2}},$$

where D is any oriented diagram for L.

For any knot K, the reverse of K, denoted by rK, is the same knot with the opposite orientation. We also denote by $\overline{K}$ the reflection of K. In general, K, rK and $\overline{K}$ are distinct knots.

Definition 2.1.7. Suppose that L is a two-component oriented link with components L_1 and L_2. The linking number $\mathrm{lk}(L_1, L_2)$ of L_1 and L_2 is half the sum of the signs, in a diagram of L, of the crossings at which one strand is from L_1 and the other from L_2.

Proposition 2.1.8. The Jones polynomial has the following properties:

1. $V(rK) = V(K)$ for any knot K,
2. if the oriented link L^* is obtained from an oriented link L by reversing the orientation of one component K, then $V(L^*) = t^{-3\,\mathrm{lk}(K,L-K)}V(L)$,
3. $V(\overline{L})$ is obtained from $V(L)$ by interchanging $t^{1/2}$ and $t^{-1/2}$.

Proof.

1. Follows from the fact that changing the orientation of a knot has no effect on the writhe (since the direction of both strands of a crossing is reversed), and that the Kauffman bracket is independent of orientation.
2. $w(D^*) = w(D) - 4\,\mathrm{lk}(K, L-K)$ (by definition of the linking number). Thus

$$\begin{aligned} V(L^*) &= (-t^{-1/4})^{-3w(D^*)}\langle D^*\rangle \\ &= (-t^{-1/4})^{-3(w(D)-4\,\mathrm{lk}(K,L-K))}\langle D\rangle \\ &= (-t^{-1/4})^{12\,\mathrm{lk}(K,L-K)}V(L) \\ &= t^{-3\,\mathrm{lk}(K,L-K)}V(L). \end{aligned}$$

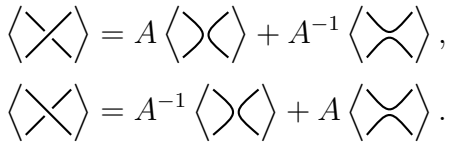
3. The reflection of a link L is equivalent to the original link with the over-crossings changed to under-crossings (and vice versa). This then follows from Lemma 2.1.3 as well as the fact that $\langle\overline{L}\rangle$ is obtained from $\langle L\rangle$ by interchanging A and A^{-1}.

□

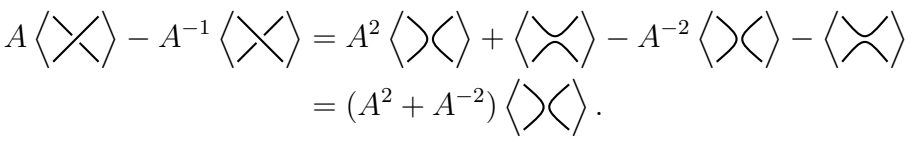
THEOREM 2.1.9. The Jones polynomial satisfies the following skein relation:

$$t^{-1}V(L_+) - tV(L_-) = (t^{1/2} - t^{-1/2})V(L_0).$$

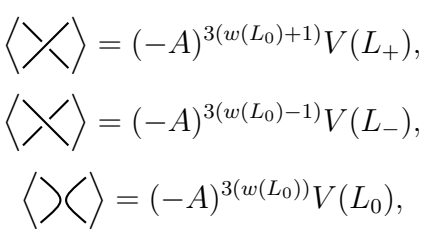
PROOF.



Hence,



Since $w(L_+) - 1 = w(L_0) = w(L_-) + 1$, and using the definition of the Jones polynomial, we have that



and hence

$$-A^4V(L_+) + A^{-4}V(L_-) = (A^2 - A^{-2})V(L_0).$$

The substitution $A^{-2} = t^{1/2}$ then gives the required result.

□

2.2 The Alexander polynomial

We now turn our attention to the second focus of this report, the Alexander polynomial. We will explore it in further detail in section 3, but we give some basic definitions and proofs here.

DEFINITION 2.2.1. For an oriented link L, we define the Alexander polynomial $\nabla_L(t)$ by the following two properties:

1. $\nabla_{\text{unknot}}(t) = 1$,
2. $\nabla_L(t)$ satisfies the skein relation

$$\nabla_{L_+} - \nabla_{L_-} = (t^{-1/2} - t^{1/2})\nabla_{L_0},$$

where L_+, L_- and L_0 differ only as shown in Figure 2.1.

Unlike the construction we gave for the Jones polynomial in the previous section, it is not immediately clear that the Alexander polynomial of a link is always defined. However, since every link has an unknotting number (see Definition 1.0.4), the calculation of the polynomial will always terminate.

DEFINITION 2.2.2. A diagram of a link is *split* if it is a disjoint union of two link diagrams. A *split link* is a link that has a split diagram.

PROPOSITION 2.2.3. If L is a split link, then $\nabla_L(t) = 0$.

PROOF. We first show that the Alexander polynomial of a trivial link with $m \geq 2$ components is 0. Let D_+, D_- and D_0 be as shown in Figure 2.2. By definition of the Alexander polynomial, we have

$$\nabla_{L_+}(t) - \nabla_{L_-}(t) = (t^{-1/2} - t^{1/2})\nabla_{L_0}(t), \tag{1}$$

where L_+, L_- and L_0 are the links represented by the diagrams D_+, D_- and D_0 respectively. Further, since D_+ and D_- are diagrams of the same link, we have

$$\nabla_{L_+}(t) = \nabla_{L_-}(t).$$

Thus, by (1), we see that $\nabla_{L_0}(t) = 0$. Since for a split diagram, the calculation of the Alexander polynomial will always terminate in a trivial link (rather than a trivial knot), we have that $\nabla_L(t) = 0$ for any link L with a split diagram.

□

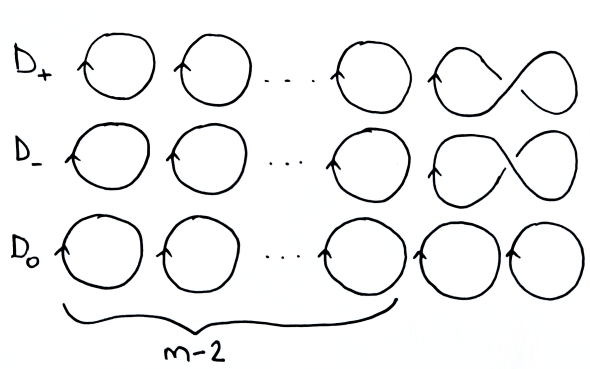


Figure 2.2: The diagrams D_+, D_- and D_0 of links L_+, L_- and L_0.

Further, we have the following proposition:

PROPOSITION 2.2.4. The Alexander polynomial has the following properties for any knot K:

1. $\nabla_{rK}(t) = \nabla_K(t)$,
2. $\nabla_{\overline{K}}(t) = \nabla_K(t)$,

We postpone a proof of Proposition 2.2.4 until section 3.2, where we reconstruct the Alexander polynomial via Seifert surfaces and matrices.

3 Invariants from Seifert surfaces

This section gives an alternate definition of the Alexander polynomial. Instead of defining it via a skein relation, we construct a surface, known as a *Seifert surface*, from a diagram of a link. From this surface we then construct a matrix, from which we recover the Alexander polynomial. This section follows the construction given in [4] (giving different examples where possible); the usual construction requires homology theory, which will not be explored in detail here (an explanation of the latter construction can be found in e.g. [3], [5]).

3.1 Seifert surfaces and matrices

THEOREM 3.1.1. Let L be a link. Then there exists an orientable, connected surface $F \subset \mathbb{R}^3$ that has as its boundary L.

PROOF. Let L be an oriented link, and D be a diagram of L. We construct an orientable, connected surface F with boundary L as follows:

First, if D is a split diagram (see Definition 2.2.2), we perform the move RII until the resulting diagram is no longer split. Then, at each crossing point in D, we perform a 'splicing operation', shown in Figure 3.1 (in the only way consistent with the orientation of the link). The resulting diagram is a union of simple closed curves. Each of these curves may now be spanned by a disc. Further, at each former crossing, we attach to the discs a half-twisted band (shown in Figure 3.2). An example of this process is shown in Figure 3.3. The resulting surface is an orientable, connected surface F with boundary L, as required.

□

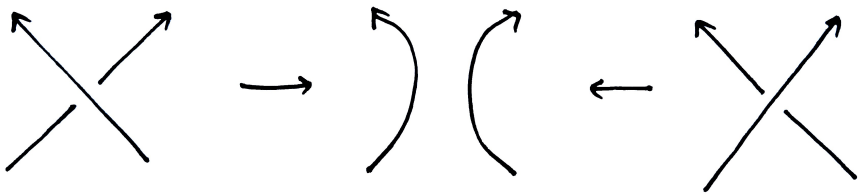


Figure 3.1: The splicing operation on positive and negative crossings.

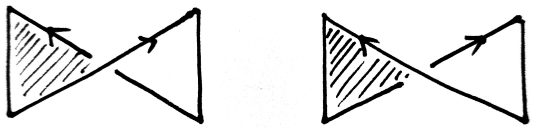


Figure 3.2: The half-twisted bands associated with positive (left) and negative (right) crossings.

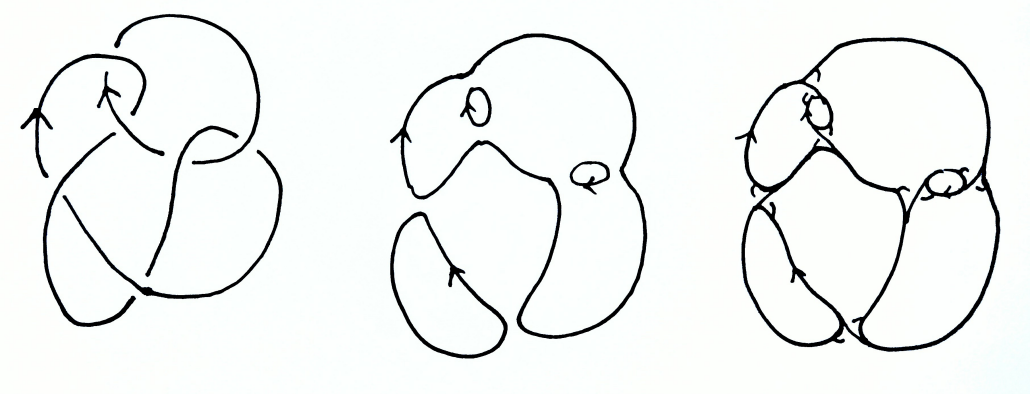


Figure 3.3: A link (L6a1), the discs obtained from splicing each crossing, and the Seifert surface of the link obtained after attaching the bands.

DEFINITION 3.1.2. Suppose a Seifert surface F is constructed from a link diagram D as in the proof of Theorem 3.1.1. The graph obtained from F by contracting each disc to a point and each band to a line is called the *Seifert graph* of D.

DEFINITION 3.1.3. The *Euler characteristic* of a surface F, denoted by $\chi(F)$, is defined by

$$\chi(F) = v - e + f, \tag{2}$$

where v, e and f are the number of vertices, edges and faces respectively in a division of F. It can be shown that $\chi(F)$ is independent of the division used, and that if $g(F)$ is the genus of F, we have

$$\chi(F) = 2 - 2g(F).$$

If F has a boundary, the above formula becomes

$$\chi(F) = 2 - m(F) - 2g(F), \tag{3}$$

where $m(F)$ is the number of closed curves that make up the boundary of F.

LEMMA 3.1.4. Let D be a diagram of an m-component link, and let F be its Seifert surface with d discs and b bands. Then

$$2g(F) + m(F) - 1 = 1 - d + b.$$

PROOF. We divide the Seifert surface as follows: the vertices are given by the four corners of each band, the edges are given by the edges of the bands, and the faces are the disks and bands. Thus, as illustrated in Figure 3.4, we have that $v = 4b, e = 6b$ and $f = b + d$. By (2), we have that $\chi(F) = 4b - 6b + b + d = d - b$, and using (3) we obtain $2g(F) + m(F) - 1 = 1 - \chi(F)$. Combining these, we see that $2g(F) + m(F) - 1 = 1 - d + b$, as required.

□

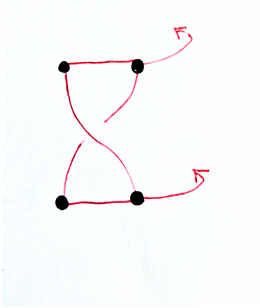


Figure 3.4: For each band in the division of F, we have four vertices (shown in black) and six edges (in red).

LEMMA 3.1.5. Let D be a diagram of an m-component link, and let $\Gamma(D)$ be its Seifert graph. Then $\Gamma(D)$ is a planar graph which partitions S^2 into $2g(F) + m(F) - 1$ domains (excluding the domain that contains ∞).

PROOF. By construction, $\Gamma(D)$ is a planar graph (since the edges are contracted bands, which come from crossings, which by definition cannot themselves cross each other). When $\Gamma(D)$ partitions S^2, there are d vertices and b edges. Thus, since the Euler characteristic of S^2 is 2, using (2), we can calculate that there are $1 - d + b$ faces (excluding the face containing ∞) in this division of S^2. Lemma 3.1.4 then gives the required result.

□

Using the result from Lemma 3.1.5, we may define $2g(F)+m(F)-1$ curves on a Seifert surface F of a link L, given by the boundaries of the domains created by the partition of S^2 by $\Gamma(D)$.[1] Figure 3.5 shows an example of this.

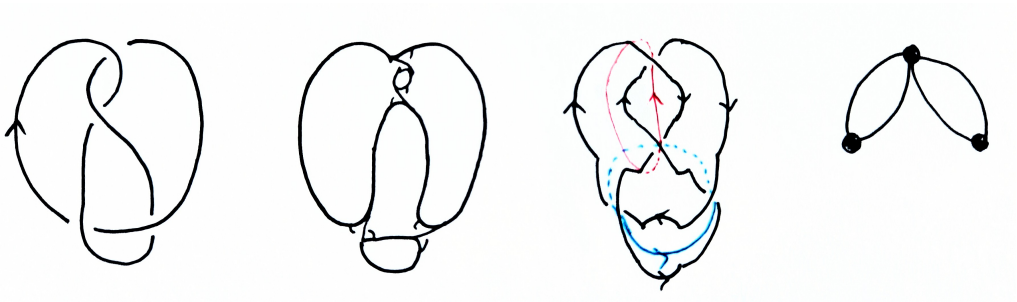


Figure 3.5: A diagram of the knot 4_1 (the 'figure-eight knot'), the Seifert surface constructed from this diagram, and its Seifert graph.

Further, for each curve α_i on F, we define the curve $\alpha_i^{\#}$ as the curve 'parallel' to α_i obtained by 'lifting' α_i slightly above F, as shown in Figure 3.6. We may assign an orientation on the curves α_i (inducing orientations on $\alpha_i^{\#}$) arbitrarily. We now define the *Seifert matrix* M as

$$M = \begin{pmatrix} \mathrm{lk}(\alpha_1, \alpha_1^{\#}) & \mathrm{lk}(\alpha_1, \alpha_2^{\#}) & \cdots & \mathrm{lk}(\alpha_1, \alpha_n^{\#}) \\ \mathrm{lk}(\alpha_2, \alpha_1^{\#}) & \mathrm{lk}(\alpha_2, \alpha_2^{\#}) & \cdots & \mathrm{lk}(\alpha_2, \alpha_n^{\#}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{lk}(\alpha_n, \alpha_1^{\#}) & \mathrm{lk}(\alpha_n, \alpha_2^{\#}) & \cdots & \mathrm{lk}(\alpha_n, \alpha_n^{\#}) \end{pmatrix},$$

where $m = 2g(F) + m(F) - 1$.

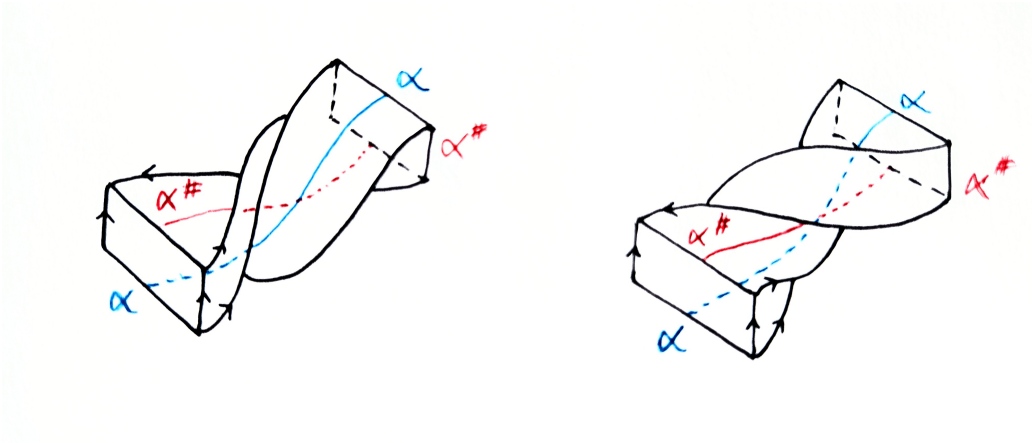


Figure 3.6: The 'lift' $\alpha^{\#}$ of α.

Example 3.1.6. We consider the knot $K = 5_1$. Using the process given in the proof of Theorem 3.1.1, we may construct a Seifert surface F of K, and identify the curves α_i on its surface, as shown in Figure 3.7.

[1] These curves are actually the generators for the homology group $H_1(F)$. An unfortunate consequence of not exploring the homology theory behind the construction of the Alexander polynomial is that they seem arbitrary when presented in this way.

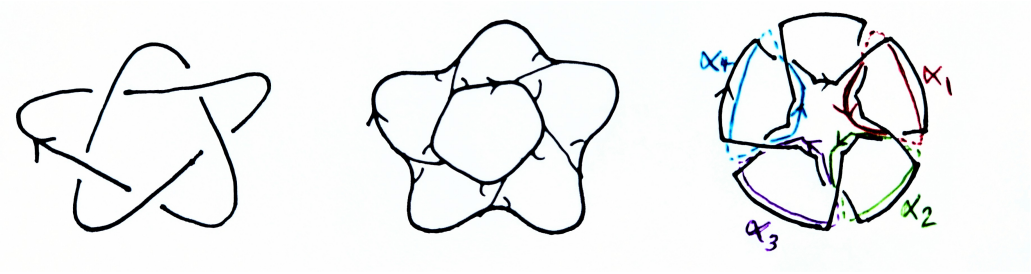


Figure 3.7: The knot 5_1, its Seifert surface, and the curves α_i.

The relationships between α_i and $\alpha_j^{\#}$ are given in Figure 3.8 (with the relationships not shown being between links which clearly have linking number 0).

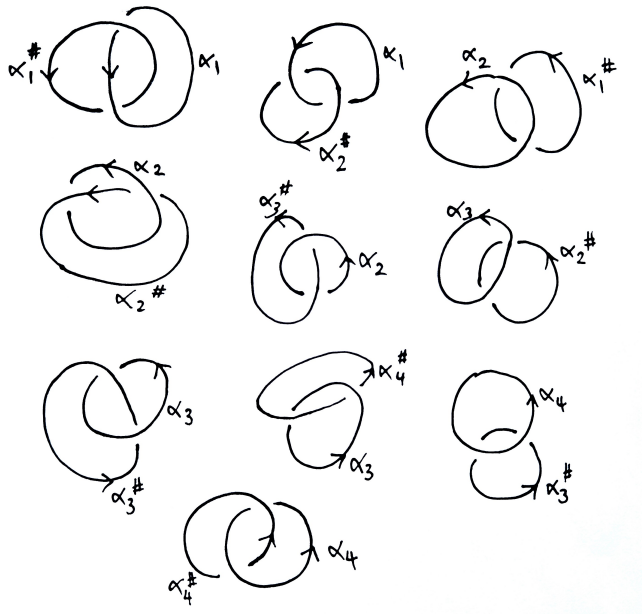


Figure 3.8: The relationships between the curves α_i.

Therefore, the Seifert matrix M is given by

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

EXAMPLE 3.1.7. Let $K = 5_2$. Figure 3.9 shows K and the Seifert surface of a diagram of K.

The curves $\alpha_1, \alpha_1^{\#}, \alpha_2$ and $\alpha_2^{\#}$ are related as shown in Figure 3.10.

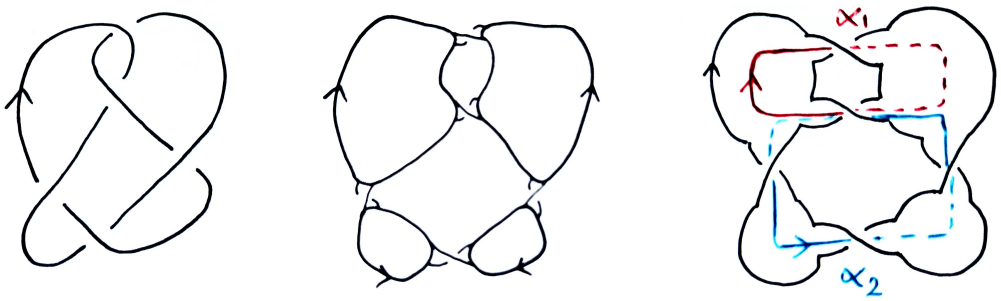


Figure 3.9: The knot 5_2, its Seifert surface, and the curves α_i.

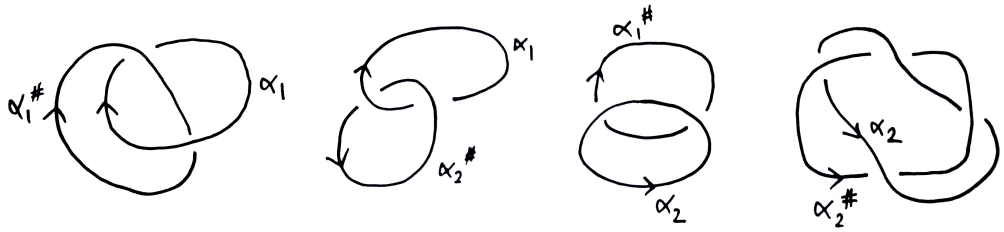


Figure 3.10: The relationships between the curves α_i

Thus, the Seifert matrix is given by

$$M = \begin{pmatrix} 1 & 1 \\ 0 & 2 \end{pmatrix}.$$

THEOREM 3.1.8. Two Seifert matrices, obtained from two equivalent links, may be changed from one to the other by applying, a finite number of times, the following two operations (and their inverses):

$$\Lambda_1 : M_1 \mapsto PM_1P^T,$$

where P is an invertible integer matrix with determinant ± 1.

$$\Lambda_2 : M_1 \mapsto \begin{pmatrix} & & & 0 & 0 \\ & M_1 & & \vdots & \vdots \\ & & & 0 & 0 \\ * & \cdots & * & 0 & 0 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix} \text{ or } \begin{pmatrix} & & & * & 0 \\ & M_1 & & \vdots & \vdots \\ & & & * & 0 \\ 0 & \cdots & 0 & 0 & 1 \\ 0 & \cdots & 0 & 0 & 0 \end{pmatrix},$$

where * denotes an arbitrary integer.

The proof of Theorem 3.1.8 is omitted here, but may be found in [4].

DEFINITION 3.1.9. Two matrices M_1, M_2 are *S-equivalent*, denoted by $M_1 \overset{\text{S}}{\sim} M_2$, if they are related by a finite sequence of the operations Λ_1, Λ_2 (and their inverses).

PROPOSITION 3.1.10. Let L be a link, rL be the link with the orientation of every component reversed, and $\overline{L}$ be the reflection of L. We denote by M_L a Seifert matrix of the link L. Then

1. $M_{rL} \overset{\text{S}}{\sim} M_L^T$

2. $M_{\overline{L}} \overset{S}{\sim} -M_L^T$

Proof.

1. If D is a diagram of L, then the diagram D', given by D with the orientations of every component reversed, is a diagram of rL. Therefore, the Seifert surfaces of D and D' have opposite orientations, and hence the over and under relations for α_i and $\alpha_j^{\#}$ are completely reversed. Thus, the Seifert matrix of D' is the transpose of the Seifert matrix of D.
2. We obtain a diagram D' of $\overline{L}$ by swapping the over- and under- segments at each crossing. Therefore, since the over and under relations for the curves α_i are completely reversed, $M_{\overline{L}} \overset{S}{\sim} -M_L^T$.

□

3.2 The Alexander polynomial

Proposition 3.2.1. If M is the Seifert matrix of a link L, then $|\det(M + M^T)|$ is an invariant of L. This invariant is the *determinant* of L.

Proof. We show that the determinant is invariant under the moves Λ_1 and Λ_2.

Let $M_1 = \Lambda_1(M) = PMP^T$, where P is an invertible integer matrix with determinant 1. Then

$$\begin{aligned}|\det(M_1 + M_1^T)| &= |\det(PMP^T + (PMP^T)^T)| \\ &= |\det(PMP^T + PM^TP^T)| \\ &= |\det(P)\det(M + M^T)\det(P^T))| \\ &= |\det(M + M^T)|.\end{aligned}$$

Further, let $M_2 = \Gamma_2(M)$. Then

$$\begin{aligned}|\det(M_2 + M_2^T)| &= \left|\det\begin{pmatrix} & & & * & 0 \\ & M + M^T & & \vdots & \vdots \\ & & & * & 0 \\ * & \cdots & * & 0 & 1 \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}\right| \\ &= |-\det(M + M^T)| \\ &= |\det(M + M^T)|.\end{aligned}$$

Thus, by Theorem 3.1.8, the determinant of a link is an invariant.

Theorem 3.2.2. Let L be a link, M be the Seifert matrix of L, and k be the order of M. Then the expression

$$t^{-k/2}\det(M - tM^T)$$

is an invariant of L.

Proof. As with Proposition 3.2.1, we show invariance under Λ_1 and Λ_2. Letting $M_1 = \Lambda_1(M)$ and $M_2 = \Lambda_2(M)$, we have

$$\begin{aligned}t^{-k/2}\det(M_1 - tM_1^T) &= t^{-k/2}\det(PMP^T - tPM^TP^T) \\ &= t^{-k/2}\det(P)\det(M - tM^T)\det(P^T) \\ &= t^{-k/2}\det(M - tM^T)\end{aligned}$$

$$t^{-(k+2)/2}\det(M_2 - tM_2^T) = t^{-1}t^{-k/2}\det\begin{pmatrix} & & & \pm b_1 & 0 \\ & M - tM^T & & \vdots & \vdots \\ & & & \pm b_k & 0 \\ \mp b_1 t & \cdots & \mp b_k t & 0 & \pm 1 \\ 0 & \cdots & 0 & \mp t & 0 \end{pmatrix}$$
$$= t(t^{-1})(t^{-k/2})\det(M - tM^T)$$
$$= t^{-k/2}\det(M - tM^T),$$

and similarly, we may verify that $t^{-(k-2)/2}\det(\Lambda_2(M) - t\Lambda_2(M)^T) = t^{-k/2}\det(M - tM^T)$.

□

THEOREM 3.2.3. The invariant defined in Theorem 3.2.2 is equal to the Alexander polynomial (defined in Definition 2.2.1).

As with Theorem 3.1.8, we omit the proof of this theorem here; it may also be found in [4].

Finally, we are now in a position to prove Proposition 2.2.4.

1. From the proof of Proposition 3.1.10, we may assume that $M_{rK} = M_K^T$. If M_K is of order k, then so is M_{rK}. Thus,

$$\begin{aligned}\nabla_{rL}(t) &= t^{-k/2}\det(M_{rK} - tM_{rK}^T) \\ &= t^{-k/2}\det(M_K^T - tM_K) \\ &= t^{-k/2}\det((M_K^T - tM_K)^T) \\ &= t^{-k/2}\det(M_K - tM_K^T) \\ &= \nabla_K(t).\end{aligned}$$

2. Similarly, we may assume that $M_{\overline{K}} = -M_K^T$. Since K is a knot, the order k of $M_{\overline{K}}$ and M_K^T must be even (it is equal to $2g(F)$). We have

$$\begin{aligned}\nabla_{\overline{K}}(t) &= t^{-k/2}\det(M_{\overline{K}} - tM_{\overline{K}}^T) \\ &= t^{-k/2}\det(-M_K^T + tM_K) \\ &= t^{-k/2}\det((-M_K^T + tM_K)^T) \\ &= t^{-k/2}\det(-M_K + tM_K^T) \\ &= (-1)^k t^{-k/2}\det(M_K - tM_K^T) \\ &= \nabla_L(t).\end{aligned}$$

□

4 Invariants of braids and tangles

The aim of this section is to rederive the Jones and Alexander polynomials by considering braids and tangles instead of links. Section 4.1 introduces braids and braid groups; as the set of isotopy classes of links can be identified with a quotient set of the union of the braid groups, we may study links via braid groups and their representations. Section 4.2 generalises these calculations to tangles. These sections (4.1 and 4.2) follow (and summarise the main ideas of) chapters 2 and 3 of [5]; some definitions and theorems are slightly rephrased, and the derivation of the Jones and Alexander polynomials from charge-conserving R-matrices is elaborated on a little further.

4.1 Braids

Definition 4.1.1. A *braid* with n strands is n oriented arcs in $\mathbb{R}^2 \times [0,1]$ such that its boundary is the set $\{1,2,...,n\} \times \{0\} \times \{0,1\}$, and such that no arc has a critical point with respect to the vertical coordinate. An example is given in Figure 4.1. Two braids b_1, b_2 are equivalent if they are ambient isotopic (the strings can be 'moved' from one position to the other, while keeping the ends fixed).

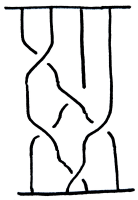


Figure 4.1: An example of a braid with 4 strands.

Definition 4.1.2. The *braid group* B_n, is the set of isotopy classes of braids with n strands, together with the operation of attaching braids vertically (see Figure 4.2).

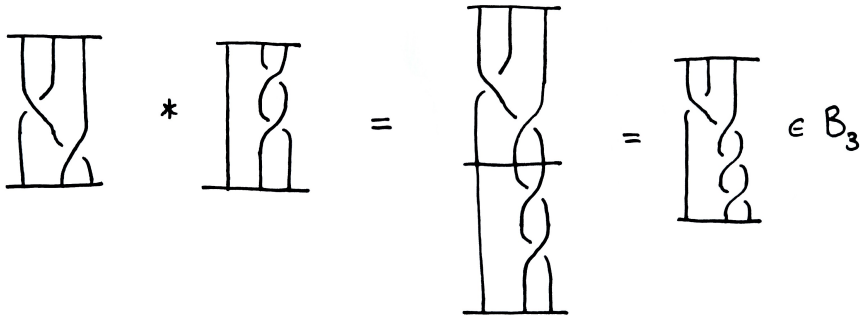


Figure 4.2: The composition of two braids in B_3.

B_n has generators $\sigma_1, \sigma_2, ..., \sigma_{n-1}$ (see Figure 4.3) and relations:

1. $\sigma_i \sigma_j = \sigma_j \sigma_i$ for $|i-j| > 2$,
2. $\sigma_i \sigma_{i+1} \sigma_i = \sigma_{i+1} \sigma_i \sigma_{i+1}$.

The *closure* of a braid is the link obtained by connecting the upper and lower ends, as shown in Figure 4.4.

We now state two important theorems which will allow us to translate the problem of link invariants into the language of braids. We omit their proofs here, but sketches of both proofs may be found in [5].

Theorem 4.1.3 (Alexander). Any oriented knot is isotopic to the closure of some (downwards oriented) braid.

Theorem 4.1.4 (Markov). Let b_1, b_2 be two braids, and L_1, L_2 their closures. Then L_1 and L_2 are isotopic if and only if b_1 and b_2 are related by a sequence of the following moves:

$$\text{MI} : ab \leftrightarrow ba, \quad a, b \in B_n,$$

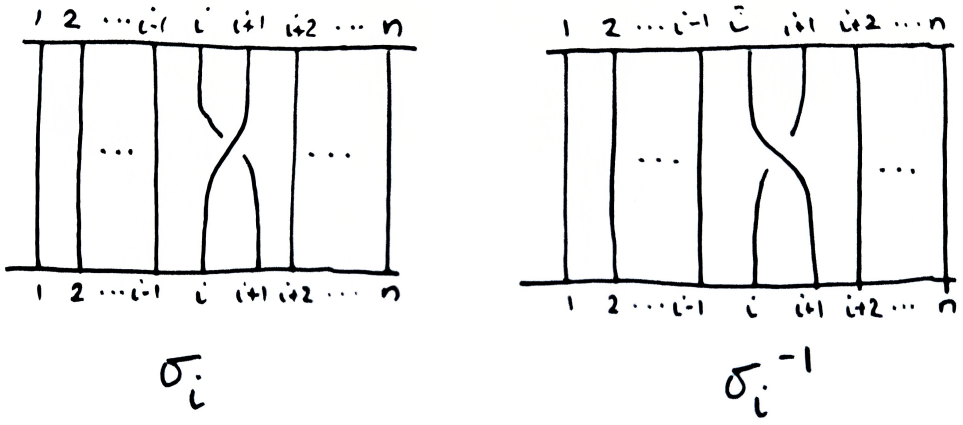


Figure 4.3: The generators σ_i (and their inverses) in the group B_n.

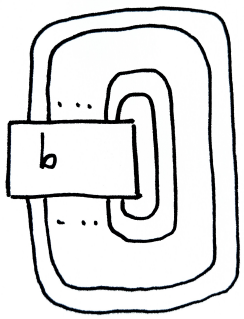


Figure 4.4: The closure of a braid b.

$$\text{MII} : b \leftrightarrow b\sigma_n^{\pm 1}, \quad b \in B_n,$$

where $b\sigma_n^{\pm 1} \in B_{n+1}$.

Theorem 4.1.4 establishes that to construct invariants of links, it is sufficient to construct properties of braids which are invariant under the Markov moves MI and MII. For the rest of section 4.1 we use the group structure of B_n to construct such link invariants.

Definition 4.1.5. Let G be a group, and let V be a vector space. A *representation* of G on V is a homomorphism $\psi : G \to \mathrm{GL}(V) \subset \mathrm{End}(V)$.

The goal of this section will be to construct a representation $\psi_n : B_n \to \mathrm{End}(V^{\otimes n})$ (associating a vector space V with each strand of a braid b), and define invariants from this representation.

We define the linear map $\tau : V \otimes V \to V \otimes V$ by $\tau(x \otimes y) = y \otimes x$ for $x, y \in V$. With respect to the basis $\{e_0 \otimes e_0, e_0 \otimes e_1, e_1 \otimes e_0, e_1 \otimes e_1\}$, τ is presented by

$$\tau = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \in \mathrm{End}(V \otimes V).$$

Our first representation of B_n is given as follows:

$$\psi_n(\sigma_i) = (\mathrm{id}_V)^{\otimes(i-1)} \otimes \tau \otimes (\mathrm{id}_V)^{\otimes(n-i-1)},$$

where τ takes on the role of a crossing, swapping the ith and $(i+1)$-th copies of V. However, it is immediately clear that ψ_n will not give rise to useful invariants; since $\tau^{-1} = \tau$, this representation cannot differentiate between over- and under-crossings. Thus, we consider a matrix $R \in \mathrm{End}(V \otimes V)$, which may be thought of as a 'perturbation' of τ, from which we may derive useful invariants. Our new representation is given by

$$\psi_n(\sigma_i) = (\mathrm{id}_V)^{\otimes(i-1)} \otimes R \otimes (\mathrm{id}_V)^{\otimes(n-i-1)}. \tag{4}$$

For ψ_n be a representation of B_n, it must satisfy:

1. $\psi_n(\sigma_i\sigma_j) = \psi_n(\sigma_j\sigma_i)$ for $|i-j| > 2$,
2. $\psi_n(\sigma_i\sigma_{i+1}\sigma_i) = \psi_n(\sigma_{i+1}\sigma_i\sigma_{i+1})$.

The first property is immediately satisfied (as $\mathrm{id}_V \otimes \mathrm{id}_V$ will commute with any element of $V \otimes V$). For ψ_n to satisfy the second property, we must have

$$(R \otimes \mathrm{id}_V)(\mathrm{id}_V \otimes R)(R \otimes \mathrm{id}_V) = (\mathrm{id}_V \otimes R)(R \otimes \mathrm{id}_V)(\mathrm{id}_V \otimes R). \tag{5}$$

Equation (5) is known as the *Yang-Baxter* equation, and its solutions are called *R-matrices*. We will only consider R-matrices which preserve the sum of subscripts of the basis - a property known as *charge conservation*, as it makes the following calculations significantly easier.[2] Under this restriction, an R-matrix R is presented by

$$R = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & b & c & 0 \\ 0 & d & e & 0 \\ 0 & 0 & 0 & f \end{pmatrix}.$$

We can then regard $V \otimes V \otimes V$ as the direct sum of the four subspaces preserved by R (the bases of which are $\{e_0 \otimes e_0 \otimes e_0\}$, $\{e_0 \otimes e_0 \otimes e_1, e_0 \otimes e_1 \otimes e_0, e_1 \otimes e_0 \otimes e_0\}$, $\{e_0 \otimes e_1 \otimes e_1, e_1 \otimes e_0 \otimes e_1, e_1 \otimes e_1 \otimes e_0\}$, and $\{e_1 \otimes e_1 \otimes e_1\}$), and calculate

$$(R \otimes \mathrm{id}_V)(\mathrm{id}_V \otimes R)(R \otimes \mathrm{id}_V) = (a^3) \oplus \begin{pmatrix} a^2b & abc & ac^2 \\ abd & b^2e + acd & bce + ace \\ ad^2 & bde + ade & cde + ae^2 \end{pmatrix} \oplus \begin{pmatrix} b^2f + bcd & bcf + bce & c^2f \\ bdf + bde & cdf + be^2 & cef \\ d^2f & def & ef^2 \end{pmatrix} \oplus (f^3),$$

$$(\mathrm{id}_V \otimes R)(R \otimes \mathrm{id}_V)(\mathrm{id}_V \otimes R) = (a^3) \oplus \begin{pmatrix} ab^2 + bcd & abc + bce & ac^2 \\ abd + bde & acd + be^2 & ace \\ ad^2 & ade & a^2e \end{pmatrix} \oplus \begin{pmatrix} bf^2 & bcf & c^2f \\ bdf & b^2e + cdf & bce + cef \\ d^2f & bde + def & cde + e^2f \end{pmatrix} \oplus (f^3).$$

Thus, for R to satisfy (5), we must have

$$b(cd + ab - a^2) = 0, \quad b(cd + bf - f^2) = 0,$$

$$e(cd + ae - a^2) = 0, \quad e(cd + ef - f^2) = 0,$$

$$bce = bde = be(b - e) = 0.$$

[2] It will turn out that this is a safe restriction to impose - when we later consider the R-matrices that arise from certain 2 and 3 dimensional representations of $U_q(\mathfrak{sl}_2)$ in section 5.4, these will satisfy charge conservation (in this case, charge conservation is equivalent to commuting with the diagonal matrix $\rho(\Delta(K))$).

In the case where $b = 0$ and $e \neq 0$, we obtain the following two R-matrices:

$$R = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & d & a - cd/a & 0 \\ 0 & 0 & 0 & a \end{pmatrix}, \quad \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & d & a - cd/a & 0 \\ 0 & 0 & 0 & -cd/a \end{pmatrix}. \tag{6}$$

4.1.1 The Jones polynomial

We consider the first of these R-matrices, and begin by defining the *partial traces* tr_1 and tr_2.

DEFINITION 4.1.6. Let V_1, V_2 be vector spaces over $\mathbb{C}$ with bases $\{e_i\}$ and $\{e'_i\}$ respectively, and let $A \in \mathrm{End}(V_1 \otimes V_2)$ be presented by $A(e_i \otimes e'_j) = \sum_{k,l} A_{ij}^{kl} e_k \otimes e'_l$. Then the partial traces $\mathrm{tr}_1(A)$ and $\mathrm{tr}_2(A)$ are given by

$$(\mathrm{tr}_1(A))_{ij} = \sum_k A_{ki}^{kj},$$

$$(\mathrm{tr}_2(A))_{ij} = \sum_k A_{ik}^{jk}.$$

We try to obtain a link invariant by taking the trace $\mathrm{tr}(\psi_n(b))$, where b is a braid in B_n - this a natural place to start because we want our invariant to have a cyclic property, as illustrated in Figure 4.5. Invariance of $\mathrm{tr}(\psi_n(b))$ under MI follows immediately from this cyclic property of the trace, however we must also have invariance under MII, given by $\mathrm{tr}_2 R^{\pm 1} = \mathrm{id}_V$ (illustrated in Figure 4.6). This does not hold for $\psi_n(b)$, and so we slightly modify our initial guess using a linear map $h \in \mathrm{End}(V)$, and consider what h must be given by to produce a link invariant.

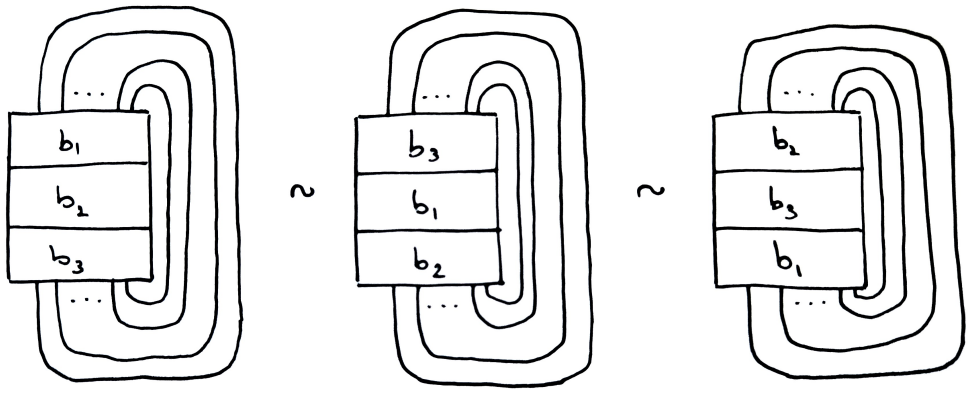


Figure 4.5: The cyclic property of braids.

Let $h \in \mathrm{End}(V)$ be given by

$$h = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix}.$$

We have

$$\mathrm{tr}_2((\mathrm{id}_V \otimes h) \cdot R) = \begin{pmatrix} h_1 a & h_2 c \\ h_3 d & h_1(a - cd/a) + h_4 a \end{pmatrix}, \tag{7}$$

$$\mathrm{tr}_2((\mathrm{id}_V \otimes h) \cdot R^{-1}) = \begin{pmatrix} h_1/a + h_4(1/a - a/(cd)) & h_2/d \\ h_3/c & h_4/a \end{pmatrix}. \tag{8}$$

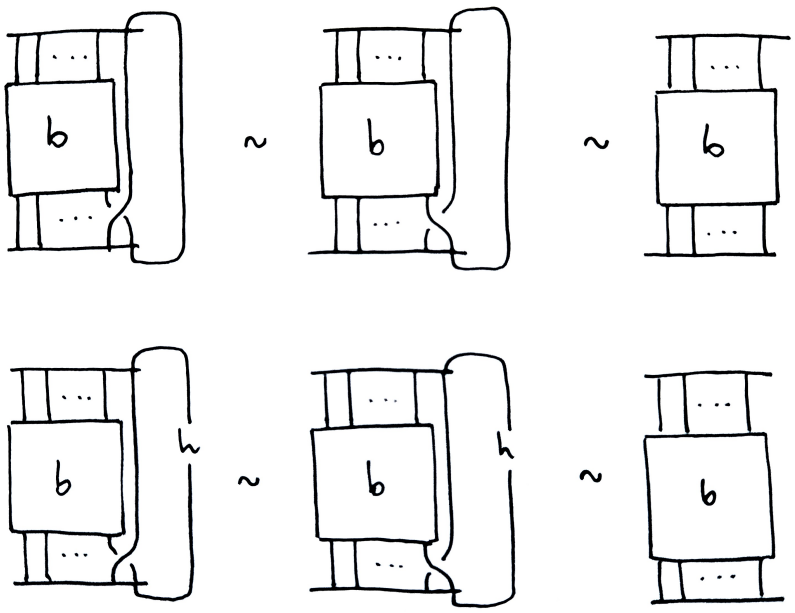


Figure 4.6: A pictorial representation of the conditions $\mathrm{tr}_2(R^{\pm 1}) = \mathrm{id}_V$ (top) and $\mathrm{tr}_2((\mathrm{id}_V \otimes h) \cdot R^{\pm 1}) = \mathrm{id}_V$ (bottom).

From (7) we see that in order for invariance under MII to hold, we must have

$$h_1 = 1/a, \quad h_4 = cd/a^3,$$

$$h_2 = h_3 = 0.$$

Further, using (8) we calculate $d = a^4/c$, and thus

$$R = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & a^4/c & a - a^3 & 0 \\ 0 & 0 & 0 & a \end{pmatrix}, \quad h = \begin{pmatrix} 1/a & 0 \\ 0 & a \end{pmatrix}.$$

THEOREM 4.1.7. Let L be an oriented link and b a braid whose closure is isotopic to L. Then, for the above representation ψ_n of B_n and the linear map h,

$$\mathrm{tr}(h^{\otimes n} \cdot \psi_n(b))$$

is invariant under the MI and MII moves. Further, under the specialisation $a^2 = c = t$, it is equal to $(t^{1/2} + t^{-1/2})$ times the Jones polynomial $V(L)$.

PROOF. We have verified above that $\mathrm{tr}_2((\mathrm{id}_V \otimes h) \cdot R^{\pm 1}) = \mathrm{id}_V$, implying that $\mathrm{tr}(h^{\otimes n} \cdot \psi_n(b))$ is invariant under MII. Further, by charge conservation of R, we have that $R \cdot (h \otimes h) = (h \otimes h) \cdot R$, and hence $\psi_n(b)$ commutes with $h^{\otimes n}$. Thus,

$$\begin{aligned} \mathrm{tr}(h^{\otimes n} \cdot \psi_n(b_1 b_2)) &= \mathrm{tr}(h^{\otimes n} \cdot \psi_n(b_1)\psi_n(b_2)) \\ &= \mathrm{tr}(\psi_n(b_2) h^{\otimes n} \cdot \psi_n(b_1)) \\ &= \mathrm{tr}(h^{\otimes n} \cdot \psi_n(b_2)\psi_n(b_1)) \\ &= \mathrm{tr}(h^{\otimes n} \cdot \psi_n(b_2 b_1)), \end{aligned}$$

demonstrating invariance under MI. Further, we show that R and R^{-1} satisfy the same skein relation as the Jones polynomial:

$$\begin{aligned} t^{-1}R - tR^{-1} &= t^{-1}\begin{pmatrix} t^{1/2} & 0 & 0 & 0 \\ 0 & 0 & t & 0 \\ 0 & t & t^{1/2}-t^{3/2} & 0 \\ 0 & 0 & 0 & t^{1/2} \end{pmatrix} - t\begin{pmatrix} t^{-1/2} & 0 & 0 & 0 \\ 0 & t^{-1/2}-t^{-3/2} & t^{-1} & 0 \\ 0 & t^{-1} & 0 & 0 \\ 0 & 0 & 0 & t^{-1/2} \end{pmatrix} \\ &= (t^{-1/2}-t^{1/2})\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= (t^{-1/2}-t^{1/2})\mathrm{id}_{V\otimes V}. \end{aligned}$$

□

4.1.2 The Alexander polynomial

We now consider the second R-matrix given in (6). As in section 4.1.1, we put $h \in \mathrm{End}(V)$ as

$$h = \begin{pmatrix} h_1 & h_2 \\ h_3 & h_4 \end{pmatrix},$$

and calculate:

$$\mathrm{tr}_2((\mathrm{id}_V \otimes h)\cdot R) = \begin{pmatrix} h_1 a & h_2 c \\ h_3 d & h_1(a-cd/a) - h_4(cd/a) \end{pmatrix}, \tag{9}$$

$$\mathrm{tr}_2((\mathrm{id}_V \otimes h)\cdot R^{-1}) = \begin{pmatrix} h_1/a + h_4(1/a - a/(cd)) & h_2/d \\ h_3/c & -h_4(a/(cd)) \end{pmatrix}. \tag{10}$$

From (9) we see that

$$h_1 = 1/a, \quad h_4 = -1/a,$$
$$h_2 = h_3 = 0.$$

Further, using (10) we calculate $d = 1/c$, and thus

$$R = \begin{pmatrix} a & 0 & 0 & 0 \\ 0 & 0 & c & 0 \\ 0 & 1/c & a-1/a & 0 \\ 0 & 0 & 0 & -1/a \end{pmatrix}, \quad h = \begin{pmatrix} 1/a & 0 \\ 0 & -1/a \end{pmatrix}.$$

Theorem 4.1.8. Let L be an oriented link and b a braid whose closure is isotopic to L. Then, for the above representation ψ_n of B_n and the linear map h,

$$\mathrm{tr}_{2,3,\ldots n}((1 \otimes h^{\otimes n-1})\psi_n(b)) = c \cdot \mathrm{id}_V,$$

and c is an isotopy invariant of L. Further, under the specialisation $a^2 = t^{-1}, c = 1$, it is equal to the Alexander polynomial $\nabla_L(t)$.

Note that the trace taken here is a partial trace; it turns out that the total trace will always vanish on this representation. We do not prove invariance here (see [5] for details), but show that R and R^{-1} satisfy the same skein relation as the Alexander polynomial:

$$\begin{aligned} R - R^{-1} &= \begin{pmatrix} t^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & t^{-1/2} - t^{1/2} & 0 \\ 0 & 0 & 0 & -t^{1/2} \end{pmatrix} - \begin{pmatrix} t^{1/2} & 0 & 0 & 0 \\ 0 & t^{1/2} - t^{-1/2} & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -t^{-1/2} \end{pmatrix} \\ &= (t^{-1/2} - t^{1/2}) \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= (t^{-1/2} - t^{1/2}) \mathrm{id}_{V \otimes V}. \end{aligned}$$

4.2 Tangles

This section generalises the earlier calculations to tangles rather than braids. We construct what are known as *operator invariants*, and rederive the Jones and Alexander polynomials as special cases of these.

Definition 4.2.1. An *oriented tangle* is a disjoint union of oriented arcs embedded in $\mathbb{R}^2 \times [0,1]$ such that the boundary is a set of distinct points in $\{0\} \times \mathbb{R} \times \{0,1\}$. An *oriented tangle diagram* is a diagram of an oriented tangle in $\mathbb{R} \times [0,1]$; any oriented tangle diagram can be expressed as a composition of tensor products of the elementary oriented tangle diagrams shown in Figure 4.7. A *sliced diagram* of an oriented tangle is an oriented tangle diagram with horizontal lines such that each domain between adjacent horizontal lines has either a single crossing or a single critical point as shown in Figure 4.8.

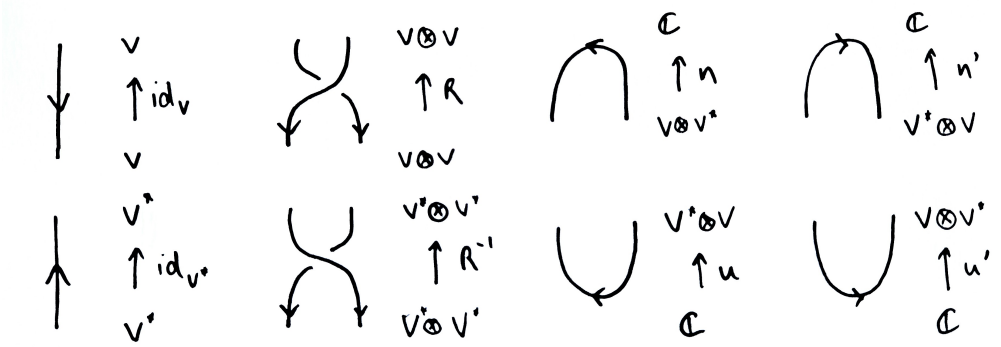


Figure 4.7: The elementary oriented tangle diagrams and the linear maps associated with them.

Theorem 4.2.2. Two oriented sliced tangle diagrams express isotopic oriented tangles if and only if the diagrams are related by a sequence of Turaev moves (shown in Figure 4.9).

Let V be a vector space with basis $\{e_i\}$, and let $\{e_i^*\}$ be the basis of V^*. We consider two invertible endomorphisms $R \in \mathrm{End}(V \otimes V)$ and $h \in \mathrm{End}(V)$. For an oriented sliced tangle diagram D with i upper ends and j lower ends, we define the bracket $[D] \in \mathrm{Hom}(V^{\otimes i}, V^{\otimes j})$ as follows. At each horizontal line, we associate the vector spaces V and V^* to downwards and upwards oriented strands respectively, and take the tensor product of the vector spaces along each such horizontal line. Each horizontal line is thus associated with a tensor product of vector spaces (horizontal lines which don't intersect the tangle are associated with $\mathbb{C}$).

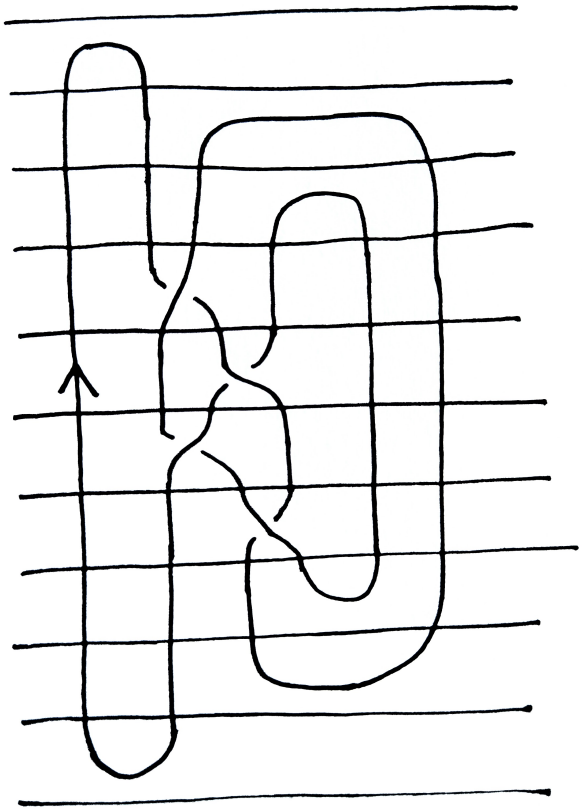


Figure 4.8: An example of an oriented sliced tangle diagram (of the knot 4_1).

Further, between each of the vector spaces associated with a horizontal line, we associate a linear map given by the tensor product of 'elementary' linear maps assigned to each elementary tangle diagram (shown in Figure 4.7). We define n and n' by $n(x \otimes f) = f(hx)$ and $n'(f \otimes x) = f(x)$ for $x \in V$, $f \in V^*$. We define u and u' by $u(1) = \sum_i e_i^* \otimes (h^{-1}e_i)$ and $u'(1) = \sum_i e_i \otimes e_i^*$.

We then define the bracket $[D] \in \mathrm{Hom}(V^{\otimes i}, V^{\otimes j})$ to be the composition of these linear maps. An example of this process is shown in Figure 4.10.

Definition 4.2.3. Let $A \in \mathrm{End}(V \otimes V)$ be such that $A(e_k \otimes e_l) = \sum_{i,j} A_{kl}^{ij} e_i \otimes e_j$. Then we define the maps $A^{\circlearrowleft} \in \mathrm{Hom}(V \otimes V^*, V^* \otimes V)$ and $A^{\circlearrowright} \in \mathrm{Hom}(V^* \otimes V, V \otimes V^*)$ by

$$A^{\circlearrowleft}(e_l \otimes e_j^*) = \sum_{k,i} A_{kl}^{ij} e_k^* \otimes e_i,$$

$$A^{\circlearrowright}(e_i^* \otimes e_k) = \sum_{j,l} A_{kl}^{ij} e_j \otimes e_l^*.$$

Theorem 4.2.4. Let T be an oriented tangle and D a sliced diagram of T. If invertible endomorphisms $R \in \mathrm{End}(V \otimes V)$ and $h \in \mathrm{End}(V)$ satisfy the following relations

1. $R \cdot (h \otimes h) = (h \otimes h) \cdot R$,
2. $\mathrm{tr}_2((\mathrm{id}_V \otimes h) \cdot R^{\pm 1}) = \mathrm{id}_V$,
3. $(R^{-1})^{\circlearrowleft} \cdot ((\mathrm{id}_V \otimes h) \cdot R \cdot (h^{-1} \otimes \mathrm{id}_V))^{\circlearrowleft} = \mathrm{id}_V \otimes \mathrm{id}_{V^*}$,
4. $(R \otimes \mathrm{id}_V)(\mathrm{id}_V \otimes R)(R \otimes \mathrm{id}_V) = (\mathrm{id}_V \otimes R)(R \otimes \mathrm{id}_V)(\mathrm{id}_V \otimes R)$,

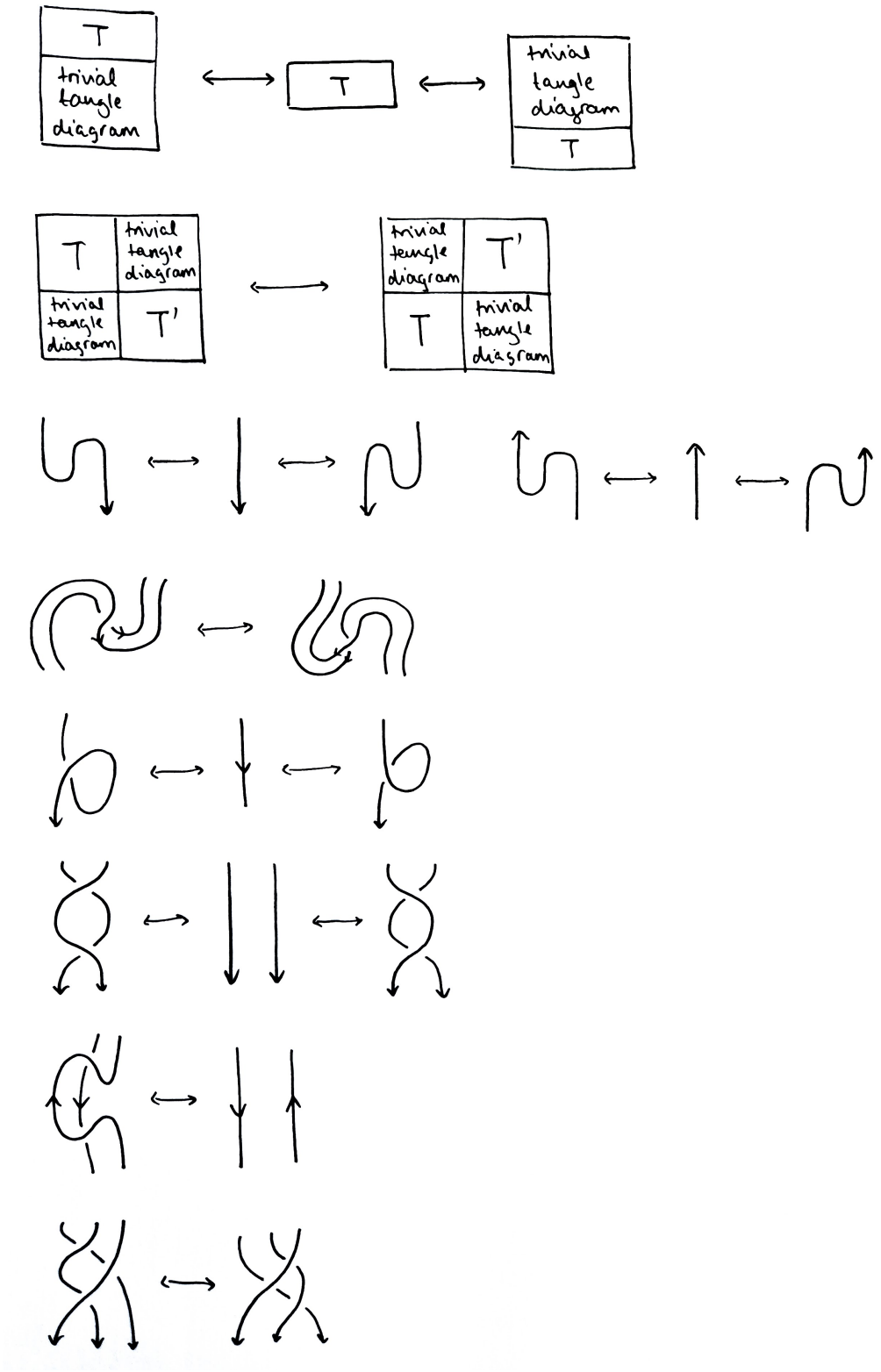


Figure 4.9: The Turaev moves for oriented tangle diagrams.

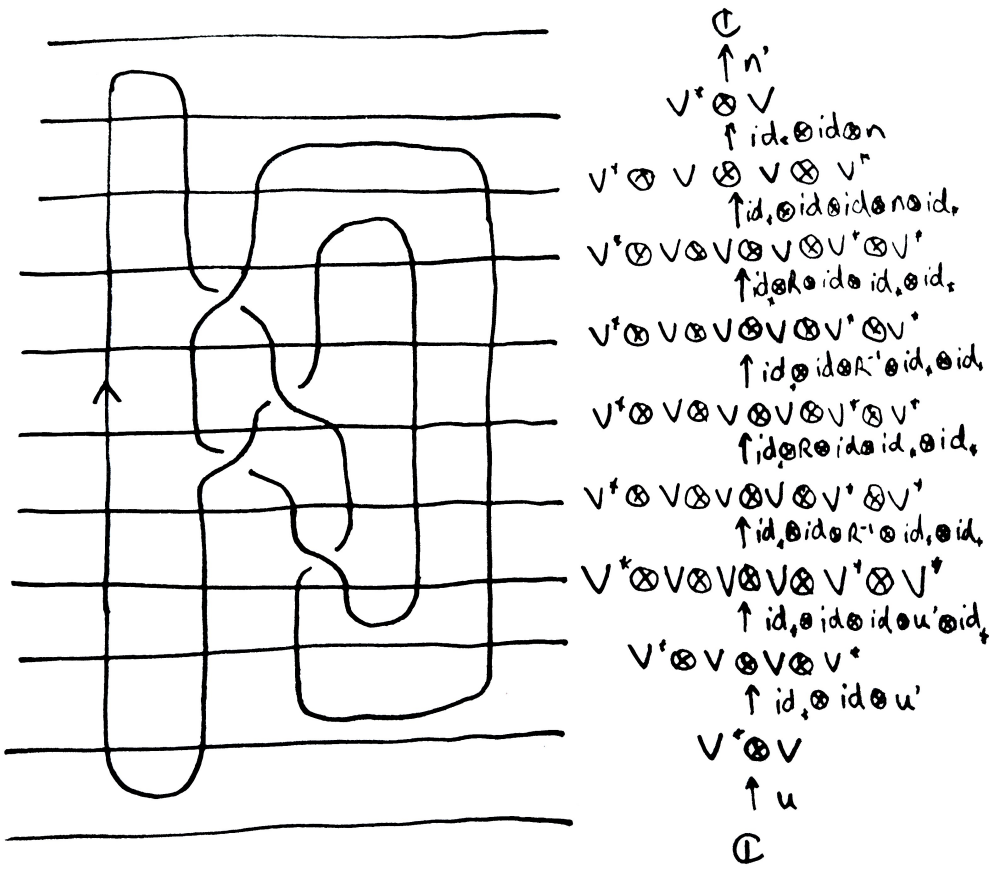


Figure 4.10: An example of the calculation of the bracket of an oriented link.

then the bracket $[D]$ is an isotopy invariant of T.

We do not give a proof of Theorem 4.2.4 here, only note that conditions $1, 2$ and 4 are the same as those required to obtain an invariant of links (given in 4.1.1), and so Theorem 4.2.4 implies that the extra condition 3 is required to give invariants of tangles.

4.2.1 The Jones polynomial

Let V be a 2-dimensional vector space over $\mathbb{C}$ with basis $\{e_0, e_1\}$. As in section 4.1.1, we put

$$R = \begin{pmatrix} t^{1/2} & 0 & 0 & 0 \\ 0 & 0 & t & 0 \\ 0 & t & t^{1/2} - t^{3/2} & 0 \\ 0 & 0 & 0 & t^{1/2} \end{pmatrix}, \quad h = \begin{pmatrix} t^{-1/2} & 0 \\ 0 & t^{1/2} \end{pmatrix}.$$

Recall that the maps n, n', u and u' are given by:

$$n(x \otimes f) = f(hx), \quad n'(f \otimes x) = f(x),$$

$$u(1) = \sum_i e_i^* \otimes (h^{-1} e_i), \quad u'(1) = \sum_i e_i \otimes e_i^*.$$

Thus, they may be represented by

$$n = \begin{pmatrix} t^{-1/2} & 0 & 0 & t^{1/2} \end{pmatrix}, \quad n' = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix},$$

$$u = \begin{pmatrix} t^{1/2} \\ 0 \\ 0 \\ t^{-1/2} \end{pmatrix}, \quad u' = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

THEOREM 4.2.5. Let L be an oriented link. The operator invariant $[L]$ from the linear maps R and h given above is equal to the Jones polynomial $V(L)$.

PROOF. In section 4.1.1 we showed that these satisfy properties 1, 2 and 4. It can further be shown that these maps satisfy property 3 (see [5] for details).

Let b be a braid with n strands whose closure is isotopic to L. By definition, we have that $[b] = \psi_n(b)$, and by construction of the operator invariant, we have

$$[L] = \mathrm{tr}(h^{\otimes n} \cdot [b]).$$

Putting $[b] = \psi_n(b)$ gives the required result.

□

4.2.2 The Alexander polynomial

Let V be a 2-dimensional vector space over $\mathbb{C}$ with basis $\{e_0, e_1\}$. As in section 4.1.2, we put

$$R = \begin{pmatrix} t^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & t^{-1/2} - t^{1/2} & 0 \\ 0 & 0 & 0 & -t^{1/2} \end{pmatrix}, \quad h = \begin{pmatrix} t^{1/2} & 0 \\ 0 & -t^{1/2} \end{pmatrix}.$$

The maps n, n', u and u' may be represented by

$$n = \begin{pmatrix} t^{1/2} & 0 & 0 & -t^{1/2} \end{pmatrix}, \quad n' = \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix},$$

$$u = \begin{pmatrix} -t^{1/2} \\ 0 \\ 0 \\ t^{1/2} \end{pmatrix}, \quad u' = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}.$$

As in section 4.2.1, it can be shown that the bracket derived from the maps R and h given above satisfies the properties in Theorem 4.2.4. However, the next proposition complicates this case slightly.

PROPOSITION 4.2.6. For any oriented link L, the operator invariant $[L]$ associated with R and h is equal to 0.

A proof is given in [5]; this proposition explains why in section 4.1.2 we had to trace over $n-1$ strands, rather than taking a total trace.

THEOREM 4.2.7. Let L be an oriented link and T a 1-tangle whose closure is isotopic to L. Then $[T]$ is an isotopy invariant, and

$$[T] = \nabla_L(t) \cdot \mathrm{id}_V.$$

5 Invariants from ribbon Hopf algebras

This section closely follows chapter 5 of [5]. We look at certain structures known as *ribbon Hopf algebras* which are particularly useful for defining link and tangle invariants. We define, in order, Hopf algebras, quasi-triangular Hopf algebras, and ribbon Hopf algebras. We then consider two (very similar) examples of ribbon Hopf algebras ($U_q(\mathfrak{sl}_2)$ and $U_\zeta(\mathfrak{sl}_2)$) which will give rise to the Jones and Alexander polynomials respectively. Relatively few proofs are given (they may be found in [5] and [2]); the aim is to give an overview on the subject. We also touch on why skein relations arise from these particular representations and ribbon Hopf algebras, and why this does not occur in general.

5.1 Hopf algebras

Definition 5.1.1. Let $\mathcal{H}$ be a vector space over $\mathbb{C}$. $\mathcal{H}$ is an *algebra* if it is equipped with a linear map $\nabla : \mathcal{H} \otimes \mathcal{H} \to \mathcal{H}$ (equivalently, a bilinear product $\nabla : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$). We write $\nabla(x \otimes y)$ as $x \cdot y$ or xy for simplicity. The algebra $\mathcal{H}$ is *associative* if the following diagram commutes:

$$\begin{array}{ccc} \mathcal{H}\otimes\mathcal{H}\otimes\mathcal{H} & \xrightarrow{\mathrm{id}\otimes\nabla} & \mathcal{H}\otimes\mathcal{H} \\ \downarrow{\scriptstyle \nabla\otimes\mathrm{id}} & & \downarrow{\scriptstyle \nabla} \\ \mathcal{H}\otimes\mathcal{H} & \xrightarrow{\nabla} & \mathcal{H} \end{array}$$

The *unit* of an algebra $\mathcal{H}$ is a homomorphism $\eta : \mathbb{C} \to \mathcal{H}$ given by $\eta(1) = 1$.

Definition 5.1.2. Let $\mathcal{H}$ be a vector space over $\mathbb{C}$. $\mathcal{H}$ is a *coalgebra* if it is equipped with a linear map $\Delta : \mathcal{H} \to \mathcal{H} \otimes \mathcal{H}$. The algebra $\mathcal{H}$ is *coassociative* if the following diagram commutes:

$$\begin{array}{ccc} \mathcal{H}\otimes\mathcal{H}\otimes\mathcal{H} & \xleftarrow[\mathrm{id}\otimes\Delta]{} & \mathcal{H}\otimes\mathcal{H} \\ {\scriptstyle \Delta\otimes\mathrm{id}}\uparrow & & \uparrow{\scriptstyle \Delta} \\ \mathcal{H}\otimes\mathcal{H} & \xleftarrow[\Delta]{} & \mathcal{H} \end{array}$$

The *counit* of a coalgebra $\mathcal{H}$ is a linear map $\epsilon : \mathcal{H} \to \mathbb{C}$ which satisfies $(\epsilon\otimes\mathrm{id})\circ\Delta = (\mathrm{id}\otimes\epsilon)\circ\Delta = \mathrm{id}$.

Definition 5.1.3. $\mathcal{H}$ is a *Hopf algebra* equipped with multiplication ∇, comultiplication Δ, unit η, counit ϵ and an antihomomorphism $S : \mathcal{H} \to \mathcal{H}$ (known as the *antipode*) if:

1. $\mathcal{H}$ is an associative algebra with multiplication ∇ and unit η,
2. $\mathcal{H}$ is a coassociative coalgebra with comultiplication Δ and counit ϵ (both of which are algebra morphisms), and
3. we have that
$$\nabla \circ (S \otimes \mathrm{id}) \circ \Delta = \eta \circ \epsilon,$$
$$\nabla \circ (\mathrm{id} \otimes S) \circ \Delta = \eta \circ \epsilon.$$

5.2 Quasi-triangular Hopf algebras

Definition 5.2.1. A *quasi-triangular Hopf algebra* is a pair $(\mathcal{H}, \mathcal{R})$, where $\mathcal{H}$ is a Hopf algebra and $\mathcal{R} \in \mathcal{H} \otimes \mathcal{H}$ is an invertible element satisfying:

1. $(\tau \circ \Delta)(x) = \mathcal{R}\Delta(x)\mathcal{R}^{-1} \quad \forall x \in \mathcal{H}$,
2. $(\Delta \otimes \mathrm{id})(\mathcal{R}) = \mathcal{R}_{13}\mathcal{R}_{23}$,

3. $(\mathrm{id} \otimes \Delta)(\mathcal{R}) = \mathcal{R}_{13}\mathcal{R}_{12}$,

where $\tau(\sum_i x_i \otimes y_i) = \sum_i y_i \otimes x_i$, and we put $\mathcal{R} = \sum_i \alpha_i \otimes \beta_i$, $\mathcal{R}_{12} = \mathcal{R} \otimes 1$, $\mathcal{R}_{23} = 1 \otimes \mathcal{R}$ and $\mathcal{R}_{13} = \sum_i \alpha_i \otimes 1 \otimes \beta_i$. We call $\mathcal{R}$ a *universal R-matrix*.

We define the element $u \in \mathcal{H}$ by $u = (\nabla \circ \tau \circ (\mathrm{id} \otimes S))(\mathcal{R}) = \sum_i S(\beta_i)\alpha_i$.

PROPOSITION 5.2.2. A quasi-triangular Hopf algebra $(\mathcal{H}, \mathcal{R})$ has the following properties:

1. $S^2(x) = uxu^{-1}$,
2. $u^{-1} = (\nabla \circ \tau \circ (\mathrm{id} \otimes S^{-1}))(\mathcal{R}^{-1}) = \sum_i S^{-1}(\beta_i')\alpha_i'$,
3. $\mathcal{R}_{12}\mathcal{R}_{13}\mathcal{R}_{23} = \mathcal{R}_{23}\mathcal{R}_{13}\mathcal{R}_{12}$,
4. $(\epsilon \otimes \mathrm{id})\mathcal{R} = 1 = (\mathrm{id} \otimes \epsilon)\mathcal{R}$,
5. $(S \otimes \mathrm{id})\mathcal{R} = \mathcal{R}^{-1} = (\mathrm{id} \otimes S^{-1})\mathcal{R}$,
6. $(S \otimes S)\mathcal{R} = \mathcal{R}$,
7. $u \otimes u \cdot \mathcal{R} = \mathcal{R} \cdot u \otimes u$,
8. $\Delta(u) = (u \otimes u) \cdot (\tau(\mathcal{R})\mathcal{R})^{-1}$,
9. $\Delta(S(u)) = (S(u) \otimes S(u)) \cdot (\tau(\mathcal{R})\mathcal{R})^{-1}$,
10. $\epsilon(u) = 1$.

5.3 Ribbon Hopf algebras

DEFINITION 5.3.1. A *ribbon Hopf algebra* is a triple $(\mathcal{H}, \mathcal{R}, v)$, where $(\mathcal{H}, \mathcal{R})$ is a quasi-triangular Hopf algebra and $v \in \mathcal{H}$ is an element satisfying:

1. v is central in $\mathcal{H}$,
2. $v^2 = S(u)u$,
3. $\Delta(v) = (v \otimes v) \cdot (\mathcal{R}_{21}\mathcal{R})^{-1}$,
4. $S(v) = v$,
5. $\epsilon(v) = 1$.

We now introduce the *universal* $\mathcal{H}$ *invariant*, $Q^{\mathcal{H};*}(L)$, of an oriented m-component link L. The values of $Q^{\mathcal{H};*}$ for elementary tangle diagrams are given in Figure 5.1. For a (framed) link diagram D, made up of elementary tangle diagrams, we construct a graphical representation of $Q^{\mathcal{H};*}(D)$ as shown in Figure 5.2. Then, for each component of D, we arbitrarily choose a point on the component, and define $Q^{\mathcal{H};*}(D) \in (\mathcal{H}/I)^{\otimes m}$ to be the product of the elements of $\mathcal{H}$ on each strand, where I is the vector subspace of $\mathcal{H}$ spanned by $xy - yx$ for any $x, y \in \mathcal{H}$ (this quotient accounts for the initial choice in the part of component to begin at).

We now construct operator invariants from the universal invariant via representations of the ribbon Hopf algebra $\mathcal{H}$.

DEFINITION 5.3.2. Let $\mathcal{H}$ be a ribbon Hopf algebra, and let V be a vector space. A *representation* of $\mathcal{H}$ on V is a homomorphism $\rho : \mathcal{H} \to \mathrm{End}(V)$. A representation is *irreducible* if no proper subspace of V is preserved by ρ.

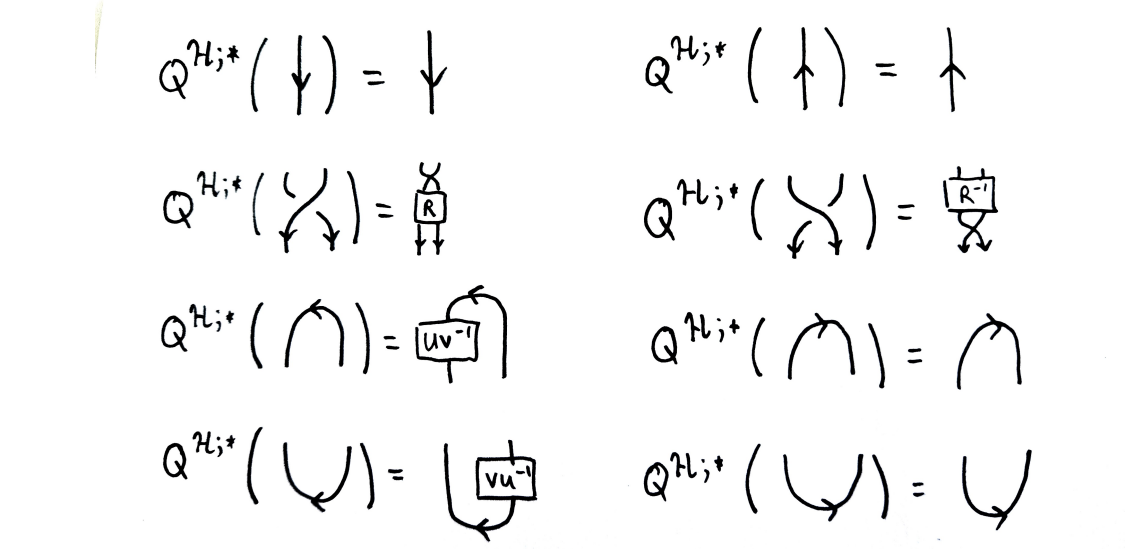


Figure 5.1: Graphical representations of the values of $Q^{\mathcal{H};*}$ on the elementary oriented tangle diagrams.

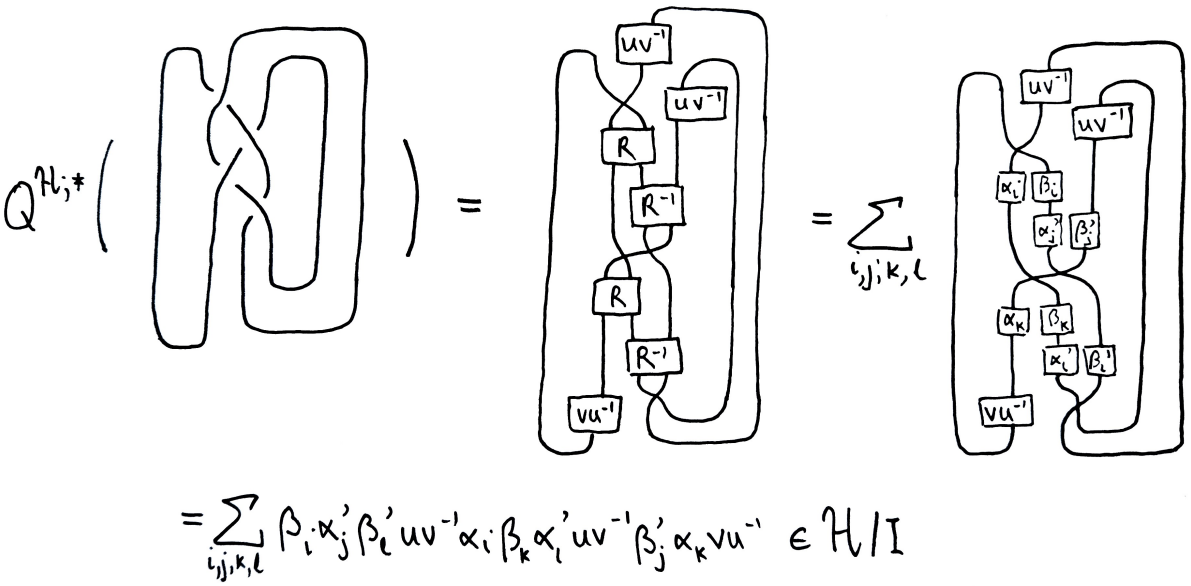


Figure 5.2: A calculation of $Q^{\mathcal{H};*}$ of an oriented link.

For an irreducible representation ρ of $\mathcal{H}$ on V, we consider the two endomorphisms, $R \in \mathrm{End}(V \otimes V)$ and $h \in \mathrm{End}(V)$ given by

$$R = \tau \circ ((\rho \otimes \rho)(\mathcal{R})), \quad h = \rho(uv^{-1}). \tag{11}$$

It can be shown (see [5] for details) that these two maps form an operator invariant, denoted by $Q^{\mathcal{H};V}$.

Definition 5.3.3. For two representations $\rho_V : \mathcal{H} \to \mathrm{End}(V)$ and $\rho_{V'} : \mathcal{H} \to V'$ of a ribbon Hopf algebra $\mathcal{H}$, the *tensor representation* $\rho_{V\otimes V'} : \mathcal{H} \to \mathrm{End}(V \otimes V')$ ($\cong \mathrm{End}(V) \otimes \mathrm{End}(V')$) of ρ_V and $\rho_{V'}$ is defined by

$$\rho_{V\otimes V'} = (\rho_V \otimes \rho_{V'}) \circ \Delta.$$

An important property of the linear map R defined in (11) is that R *commutes with the action of* $\mathcal{H}$. That is, the diagram below commutes for any $x \in \mathcal{H}$.

$$\begin{array}{ccc} V \otimes V & \xrightarrow{\rho_{V\otimes V}(x)} & V \otimes V \\ \downarrow R & & \downarrow R \\ V \otimes V & \xrightarrow{\rho_{V\otimes V}(x)} & V \otimes V \end{array}$$

This follows from the first property of Definition 5.2.1: for any $x \in \mathcal{H}$, we have

$$(\tau \circ \Delta)(x)\mathcal{R} = \mathcal{R}\Delta(x).$$

Thus, since $\rho_V \otimes \rho_V$ is a homomorphism, we have

$$((\rho_V \otimes \rho_V) \circ \tau \circ \Delta)(x) \cdot (\rho_V \otimes \rho_V)(\mathcal{R}) = (\rho_V \otimes \rho_V)(\mathcal{R}) \cdot ((\rho_V \otimes \rho_V) \circ \Delta)(x).$$

Composing τ with both sides gives

$$((\rho_V \otimes \rho_V) \circ \Delta)(x) \cdot (\tau \circ (\rho_V \otimes \rho_V))(\mathcal{R}) = (\tau \circ (\rho_V \otimes \rho_V))(\mathcal{R}) \cdot ((\rho_V \otimes \rho_V) \circ \Delta)(x),$$

which, by definition of the linear map R and the tensor representation $\rho_{V\otimes V}$, is equivalent to

$$(\rho_{V\otimes V})(x) \cdot R = R \cdot (\rho_{V\otimes V})(x).$$

In general, for two representations ρ_V and ρ_W (of a ribbon Hopf algebra $\mathcal{H}$), a linear map $f : V \to W$ is called an *intertwiner* if the following diagram commutes for any $x \in \mathcal{H}$.

$$\begin{array}{ccc} V & \xrightarrow{\rho_V(x)} & V \\ \downarrow f & & \downarrow f \\ W & \xrightarrow{\rho_W(x)} & W \end{array}$$

We have shown above that R is an intertwiner (with respect to the action of $\mathcal{H}$ on $V \otimes V$).[3]

[3] By considering the maps n and n', it can actually be shown that the entire operator invariant $Q^{\mathcal{H};V}(T)$ of a tangle T is intertwiner with respect to the action of $\mathcal{H}$ (see [5]). We only do the calculation for R here because it is referred to later.

5.4 The algebra $U_q(\mathfrak{sl}_2)$ at a generic q

We now turn to our first example of a ribbon Hopf algebra, which will give rise to the Jones polynomial.

$U_q(\mathfrak{sl}_2)$ is the algebra over $\mathbb{C}$, with unit element 1, generated by K, K^{-1}, E, F and subject to the following relations:

1. $K \cdot K^{-1} = K^{-1} \cdot K = 1$,
2. $KE = qEK$,
3. $KF = q^{-1}FK$,
4. $EF - FE = \frac{K - K^{-1}}{q^{1/2} - q^{-1/2}}$,

where $q \in \mathbb{C}$ is not a root of unity.

We can introduce a Hopf algebra structure on $U_q(\mathfrak{sl}_2)$ by defining the following:

$$\begin{aligned}
&\Delta(K^{\pm 1}) = K^{\pm 1} \otimes K^{\pm 1}, \quad S(K^{\pm 1}) = K^{\mp 1}, \quad \epsilon(K^{\pm 1}) = 1, \\
&\Delta(E) = E \otimes K + 1 \otimes E, \quad S(E) = -EK^{-1}, \quad \epsilon(E) = 0, \\
&\Delta(F) = F \otimes 1 + K^{-1} \otimes F, \quad S(F) = -KF, \quad \epsilon(F) = 0.
\end{aligned}$$

We define the *quantum integer* $[n] := (q^{n/2} - q^{-n/2})/(q^{1/2} - q^{-1/2})$, and the *quantum factorial* $[n]! := [n][n-1]...[1]$. The Hopf algebra $U_q(\mathfrak{sl}_2)$ becomes a quasi-triangular Hopf algebra when given the universal R-matrix

$$\mathcal{R} = q^{H \otimes H/4} \exp_q((q^{1/2} - q^{-1/2}) E \otimes F),$$

where H is such that $K = q^{H/2}$, and the map $\exp_q$ as defined as

$$\exp_q(x) = \sum_{n=0}^{\infty} \frac{q^{n(n-1)/4}}{[n]!} x^n.$$

Further, the elements u and v are given by

$$u = q^{-H^2/4} \sum_{n=0}^{\infty} q^{3n(n-1)/4} \frac{(q^{-1/2} - q^{1/2})^n}{[n]!} F^n K^{-n} E^n,$$

$$v = K^{-1}u = q^{-H^2/4} \sum_{n=0}^{\infty} q^{3n(n-1)/4} \frac{(q^{-1/2} - q^{1/2})^n}{[n]!} F^n K^{-n-1} E^n.$$

The triple $(U_q(\mathfrak{sl}_2), \mathcal{R}, v)$ forms a ribbon Hopf algebra with the following irreducible n-dimensional representations:

$$\rho_{V_n}(E) = \begin{pmatrix} 0 & [n-1] & & & 0 \\ & 0 & [n-2] & & \\ & & \ddots & \ddots & \\ & & & 0 & [1] \\ 0 & & & & 0 \end{pmatrix}, \quad \rho_{V_n}(F) = \begin{pmatrix} 0 & & & & 0 \\ [1] & 0 & & & \\ & [2] & 0 & & \\ & & \ddots & \ddots & \\ 0 & & & [n-1] & 0 \end{pmatrix}, \quad (12)$$

$$\rho_{V_n}(K) = \begin{pmatrix} q^{(n-1)/2} & & & & 0 \\ & q^{(n-3)/2} & & & \\ & & q^{(n-5)/2} & & \\ & & & \ddots & \\ 0 & & & & q^{-(n-1)/2} \end{pmatrix}. \tag{13}$$

5.4.1 The Jones polynomial

Setting $n = 2$ in the representations given in (12) and (13) gives

$$\rho_{V_2}(E) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \quad \rho_{V_2}(F) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad \rho_{V_2}(K) = \begin{pmatrix} q^{1/2} & 0 \\ 0 & q^{-1/2} \end{pmatrix}.$$

We calculate (noting that $\rho_{V_2}(E^2) = \rho_{V_2}(F^2) = 0$, and using that ρ_{V_2} is a homomorphism)

$$R = P \circ (\rho_{V_2} \otimes \rho_{V_2})(\mathcal{R}) = \begin{pmatrix} q^{1/4} & 0 & 0 & 0 \\ 0 & 0 & q^{-1/4} & 0 \\ 0 & q^{-1/4} & q^{1/4} - q^{-3/4} & 0 \\ 0 & 0 & 0 & q^{1/4} \end{pmatrix}.$$

Further, we compute $\rho_{V_2}(v)$

$$\rho_{V_2}(v) = \begin{pmatrix} q^{-3/4} & 0 \\ 0 & q^{-3/4} \end{pmatrix} = q^{3/4} \cdot \mathrm{id}_{V_2}.$$

Recall that the ribbon element v represented a 'twist' in a framed link. To obtain an invariant of unframed links, we normalise the R-matrix given above (similar to how we normalised the Kauffman bracket polynomial by writhe in Definition 2.1.6).

$$q^{-3/4}R = \begin{pmatrix} q^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & q^{-1} & 0 \\ 0 & q^{-1} & q^{-1/2} - q^{-3/2} & 0 \\ 0 & 0 & 0 & q^{-1/2} \end{pmatrix}.$$

We also calculate h:

$$h = \rho_{V_2}(uv^{-1}) = \rho_{V_2}(K) = \begin{pmatrix} q^{1/2} & 0 \\ 0 & q^{-1/2} \end{pmatrix}.$$

Under the substitution $q = t^{-1}$, the normalised R and h are given by

$$\begin{pmatrix} t^{1/2} & 0 & 0 & 0 \\ 0 & 0 & t & 0 \\ 0 & t & t^{1/2} - t^{3/2} & 0 \\ 0 & 0 & 0 & t^{1/2} \end{pmatrix} \text{ and } \begin{pmatrix} t^{-1/2} & 0 \\ 0 & t^{1/2} \end{pmatrix}$$

respectively, recovering the endomorphisms associated with the Jones polynomial (given in section 4.1.1).

One might wonder, at this point, whether higher dimensional representations of $U_q(\mathfrak{sl}_2)$ also give rise skein relations. However, it can be shown (using e.g. the functions defined in A.3) that

this is not the case (at least, for the 3-dimensional representations). In fact, the 2-dimensional representations considered here are a special case; we showed in section 5.3 that the R-matrix must commute with the action of the Hopf algebra (here, $U_q(\mathfrak{sl}_2)$), and in fact in A.3.1 we show that the dimension of the subspace of $\mathrm{End}(V)$ spanned by matrices which commute with the action of $U_q(\mathfrak{sl}_2)$ is 2-dimensional. Therefore, R, R^{-1} and id_V are forced to satisfy a skein relation. In higher dimensions, the subspace spanned by matrices commuting with the action of $U_q(\mathfrak{sl}_2)$ is larger (e.g. for the 3-dimensional representations, the subspace is 3-dimensional).

5.5 The algebra $U_\zeta(\mathfrak{sl}_2)$ at a root of unity ζ

The second ribbon Hopf algebra we consider is a slight variation on $U_q(\mathfrak{sl}_2)$, and will give rise to the Alexander polynomial.

$U_\zeta(\mathfrak{sl}_2)$ is the algebra over $\mathbb{C}$, with unit element 1, generated by K, K^{-1}, E, F and subject to the following relations:

1. $K \cdot K^{-1} = K^{-1} \cdot K = 1$,
2. $KE = \zeta EK$,
3. $KF = \zeta^{-1} FK$,
4. $EF - FE = \frac{K - K^{-1}}{\zeta^{1/2} - \zeta^{-1/2}}$,

where $\zeta \in \mathbb{C}$ *is* a root of unity ($\zeta^r = 1$ for some $r \in \mathbb{C}$). We define the universal R-matrix as

$$\mathcal{R} = \zeta^{H \otimes H/4} \exp_\zeta^{(<r)}((\zeta^{1/2} - \zeta^{-1/2}) E \otimes F),$$

where the map $\exp_q^{(<r)}$ is defined as

$$\exp_q^{(<r)}(x) = \sum_{0 \le n < r} \frac{q^{n(n-1)/4}}{[n]!} x^n.$$

The elements u and v are given by

$$u = \zeta^{-H^2/4} \sum_{n=0}^{r-1} \zeta^{3n(n-1)/4} \frac{(\zeta^{-1/2} - \zeta^{1/2})^n}{[n]!} F^n K^{-n} E^n,$$

$$v = K^{r-1} u = \zeta^{-H^2/4} \sum_{n=0}^{\infty} \zeta^{3n(n-1)/4} \frac{(\zeta^{-1/2} - \zeta^{1/2})^n}{[n]!} F^n K^{r-n-1} E^n.$$

$(U_\zeta(\mathfrak{sl}_2), \mathcal{R}, v)$ has the following irreducible n-dimensional representations:

$$\rho_{r,\lambda}(E) = \begin{pmatrix} 0 & [r-1+\lambda] & & & 0 \\ & 0 & [r-2+\lambda] & & \\ & & \ddots & \ddots & \\ & & & 0 & [1+\lambda] \\ 0 & & & & 0 \end{pmatrix}, \quad \rho_{r,\lambda}(F) = \begin{pmatrix} 0 & & & & 0 \\ [1] & 0 & & & \\ & [2] & 0 & & \\ & & \ddots & \ddots & \\ 0 & & & [r-1] & 0 \end{pmatrix},$$

$$\rho_{r,\lambda}(K) = \begin{pmatrix} \zeta^{(r-1+\lambda)/2} & & & & 0 \\ & \zeta^{(r-3+\lambda)/2} & & & \\ & & \zeta^{(r-5+\lambda)/2} & & \\ & & & \ddots & \\ 0 & & & & \zeta^{(-r+1+\lambda)/2} \end{pmatrix},$$

where $\lambda \in \mathbb{C}$ is a complex parameter.

5.5.1 The Alexander polynomial

Setting $r = 2$ in the above representations gives

$$\rho_\lambda(E) = \begin{pmatrix} 0 & [1+\lambda] \\ 0 & 0 \end{pmatrix}, \quad \rho_\lambda(F) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad \rho_\lambda(K) = \begin{pmatrix} \zeta^{(1+\lambda)/2} & 0 \\ 0 & \zeta^{(-1+\lambda)/2} \end{pmatrix}, \tag{14}$$

where we have fixed $\zeta = -1$, and $\log(\zeta) = i\pi$ (in order to ensure that the complex powers in (14) are uniquely defined). We further calculate (noting that $\zeta^2 = 1$),

$$R = P \circ (\rho_\lambda \otimes \rho_\lambda)(\mathcal{R}) = \zeta^{(\lambda^2-1)/4} \begin{pmatrix} \zeta^{(\lambda+1)/2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & \zeta^{(1+\lambda)/2} - \zeta^{-(1+\lambda)/2} & 0 \\ 0 & 0 & 0 & \zeta^{(1-\lambda)/2} \end{pmatrix},$$

$$\rho_\lambda(v) = \begin{pmatrix} \zeta^{(1-\lambda^2)/4} & 0 \\ 0 & \zeta^{(1-\lambda^2)/4} \end{pmatrix} = \zeta^{(1-\lambda^2)/4} \cdot \mathrm{id}_{V_2}.$$

As in section 5.4.1, we normalise the R-matrix to obtain

$$\zeta^{(1-\lambda^2)/4} R = \begin{pmatrix} t^{-1/2} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & t^{-1/2} - t^{1/2} & 0 \\ 0 & 0 & 0 & -t^{1/2} \end{pmatrix},$$

with $t^{1/2} = \zeta^{-(1+\lambda)/2}$. Further (factoring out $\zeta = -1$),

$$h = \rho_\lambda(uv^{-1}) = \rho_\lambda(K^{-1}) = \begin{pmatrix} t^{1/2} & 0 \\ 0 & -t^{1/2} \end{pmatrix}.$$

Again, these are the matrices obtained in section 4.1.2 for the Alexander polynomial.

6 Invariants from ribbon Hopf superalgebras

In the final section, we turn our attention to ribbon Hopf superalgebras, a slight variation on the ribbon Hopf algebras we considered in section 5. The main difference will be the concept of *even* and *odd* elements (and maps); the definitions given below will be similar to those given in section 5, with the addition of extra signs to account for this. Definitions and properties are mostly from [6], further detail on Hopf superalgebras can be found here (our focus will be on using ribbon Hopf superalgebras to define link and tangle invariants, rather than exploring the more general Hopf superalgebras).

6.1 Ribbon Hopf superalgebras

Definition 6.1.1. A *super vector space* is a $\mathbb{Z}_2$-graded vector space with decomposition $V = V_0 \oplus V_1$. Elements of V_0 or V_1 are called *homogeneous*. The *parity* of a homogeneous element $x \in V$, denoted by $|x|$, is 0 for $x \in V_0$ (x *even*) and 1 for $x \in V_1$ (x *odd*).

Definition 6.1.2. A *superalgebra* over $\mathbb{C}$ is a $\mathbb{Z}_2$-graded algebra $\mathbb{C}$. That is, if $\mathcal{H}$ is a super vector space over $\mathbb{C}$, then $\mathcal{H}$ is a *superalgebra* if it is equipped with a linear map $\nabla : \mathcal{H} \otimes \mathcal{H} \to \mathcal{H}$ (equivalently, a bilinear product $\nabla : \mathcal{H} \times \mathcal{H} \to \mathcal{H}$) with $\mathcal{H}_i \mathcal{H}_j \subseteq \mathcal{H}_{i+j}$ (modulo 2).

The unit of a superalgebra $\mathcal{H}$ is a linear map $\eta : \mathbb{C} \to \mathcal{H}$ given by $\eta(1) = 1$.

If A, B are superalgebras then so is $A \otimes B$, with product given by $(w \otimes x)(y \otimes z) = (-1)^{|x||y|} wy \otimes xz$. If A, B, C, D are superalgebras and $f : A \to C$, $g : B \to D$ are linear maps, then $f \otimes g : A \otimes B \to C \otimes D$ is given by $(f \otimes g)(x \otimes y) = (-1)^{|x||g|} f(x) \otimes g(y)$.

Definition 6.1.3. Let $\mathcal{H}$ be a super vector space over $\mathbb{C}$. $\mathcal{H}$ is a *supercoalgebra* if it is equipped with an even linear map $\Delta : \mathcal{H} \to \mathcal{H} \otimes \mathcal{H}$.

The counit of a supercoalgebra $\mathcal{H}$ is an even linear map $\epsilon : \mathcal{H} \to \mathbb{C}$ which satisfies $(\epsilon \otimes \mathrm{id}) \circ \Delta = (\mathrm{id} \otimes \epsilon) \circ \Delta = \mathrm{id}$.

Definition 6.1.4. $\mathcal{H}$ is a *Hopf superalgebra* equipped with multiplication ∇, comultiplication Δ, unit η, counit ϵ and antipode $S : \mathcal{H} \to \mathcal{H}$ if:

1. $\mathcal{H}$ is an associative superalgebra with multiplication ∇, and unit η,
2. $\mathcal{H}$ is a coassociative supercoalgebra with comultiplication Δ and counit ϵ (both of which are superalgebra morphisms), and
3. we have that
$$\nabla \circ (S \otimes \mathrm{id}) \circ \Delta = \eta \circ \epsilon,$$
$$\nabla \circ (\mathrm{id} \otimes S) \circ \Delta = \eta \circ \epsilon.$$

Note that for S to satisfy the final two axioms, we must have that $S(xy) = (-1)^{|x||y|} S(y)S(x)$, contrasting the case for a Hopf algebra.

Definition 6.1.5. A *quasi-triangular Hopf superalgebra* is a pair $(\mathcal{H}, \mathcal{R})$, where $\mathcal{H}$ is a Hopf superalgebra and $\mathcal{R} \in \mathcal{H} \otimes \mathcal{H}$ is an invertible element satisfying:

1. $(\tau \circ \Delta)(x) = \mathcal{R} \Delta \mathcal{R}^{-1} \quad \forall x \in \mathcal{H}$,
2. $(\Delta \otimes \mathrm{id})(\mathcal{R}) = \mathcal{R}_{13} \mathcal{R}_{23}$,
3. $(\mathrm{id} \otimes \Delta)(\mathcal{R}) = \mathcal{R}_{13} \mathcal{R}_{12}$,

where $\tau(\sum_i x_i \otimes y_i) = \sum_i (-1)^{|x_i||y_i|} y_i \otimes x_i$.

6.2 The superalgebra $\mathfrak{psl}(1|1)$

The focus of this section will be the *projective special linear Lie superalgebra* of rank $1|1$, denoted by $\mathfrak{psl}(1|1)$. It is defined as the algebra $\mathcal{H} = \mathbb{C}[\psi_+, \psi_-]$, with relations

$$\psi_\pm^2 = 0, \quad \{\psi_+, \psi_-\} := \psi_+ \psi_- + \psi_- \psi_+ = 0,$$

$$|\psi_\pm| \equiv 1 \mod 2, \quad |1| \equiv 0 \mod 2,$$

and with comultiplication, counit and antipode given by

$$\Delta(\psi_\pm) = \psi_\pm \otimes 1 + 1 \otimes \psi_\pm,$$
$$\epsilon(\psi_\pm) = 0,$$
$$S(\psi_\pm) = -\psi_\pm.$$

It can further be equipped with universal R-matrix

$$\mathcal{R} = \exp(-\psi_+ \otimes \psi_-) = 1 - \psi_+ \otimes \psi_-.$$

We verify that $\mathcal{H}$ is a quasi-triangular Hopf superalgebra:

1. $(\Delta \otimes \mathrm{id}) \circ \Delta = (\mathrm{id} \otimes \Delta) \circ \Delta$

$$\begin{aligned}(\Delta \otimes \mathrm{id}) \circ \Delta(\psi_\pm) &= (\Delta \otimes \mathrm{id})(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= (\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \otimes 1 + 1 \otimes 1 \otimes \psi_\pm \\ &= \psi_\pm \otimes 1 \otimes 1 + 1 \otimes \psi_\pm \otimes 1 + 1 \otimes 1 \otimes \psi_\pm.\end{aligned}$$

$$\begin{aligned}(\mathrm{id} \otimes \Delta) \circ \Delta(\psi_\pm) &= (\mathrm{id} \otimes \Delta)(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= \psi_\pm \otimes 1 \otimes 1 + 1 \otimes (\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= \psi_\pm \otimes 1 \otimes 1 + 1 \otimes \psi_\pm \otimes 1 + 1 \otimes 1 \otimes \psi_\pm.\end{aligned}$$

2. $(\epsilon \otimes \mathrm{id}) \circ \Delta = \mathrm{id} = (\mathrm{id} \otimes \epsilon) \circ \Delta$

$$\begin{aligned}(\epsilon \otimes \mathrm{id}) \circ \Delta(\psi_\pm) &= (\epsilon \otimes \mathrm{id})(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= 0 + \psi_\pm = \psi_\pm.\end{aligned}$$

$$\begin{aligned}(\mathrm{id} \otimes \epsilon) \circ \Delta(\psi_\pm) &= (\mathrm{id} \otimes \epsilon)(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= \psi_\pm + 0 = \psi_\pm.\end{aligned}$$

3. $\nabla \circ (S \otimes \mathrm{id}) \circ \Delta = \eta \circ \epsilon = \nabla \circ (\mathrm{id} \otimes S) \circ \Delta$

$$\begin{aligned}\nabla \circ (S \otimes \mathrm{id}) \circ \Delta(\psi_\pm) &= \nabla \circ (S \otimes \mathrm{id})(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= \nabla(-\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= -\psi_\pm + \psi_\pm = 0.\end{aligned}$$

$$\begin{aligned}\nabla \circ (\mathrm{id} \otimes S) \circ \Delta(\psi_\pm) &= \nabla \circ (\mathrm{id} \otimes S)(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= \nabla(\psi_\pm \otimes 1 - 1 \otimes \psi_\pm) \\ &= \psi_\pm - \psi_\pm = 0.\end{aligned}$$

4. $(\tau \circ \Delta)(x)\mathcal{R} = \mathcal{R}\Delta(x) \quad \forall x \in \mathcal{H}$

$$\begin{aligned}(\tau \circ \Delta)(\psi_\pm)\mathcal{R} &= (\tau(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm))(1 \otimes 1 - \psi_+ \otimes \psi_-) \\ &= (1 \otimes \psi_\pm + \psi_\pm \otimes 1)(1 \otimes 1 - \psi_+ \otimes \psi_-) \\ &= 1 \otimes \psi_\pm + \psi_+ \otimes \psi_\pm\psi_- + \psi_\pm \otimes 1 - \psi_\pm\psi_+ \otimes \psi_-.\end{aligned}$$

$$\begin{aligned}\mathcal{R}\Delta(\psi_\pm) &= (1 \otimes 1 - \psi_+ \otimes \psi_-)(\psi_\pm \otimes 1 + 1 \otimes \psi_\pm) \\ &= \psi_\pm \otimes 1 + 1 \otimes \psi_\pm + \psi_+\psi_\pm \otimes \psi_- - \psi_+ \otimes \psi_-\psi_\pm \\ &= 1 \otimes \psi_\pm + \psi_+ \otimes \psi_\pm\psi_- + \psi_\pm \otimes 1 - \psi_\pm\psi_+ \otimes \psi_-.\end{aligned}$$

5. $(\Delta \otimes \mathrm{id})(\mathcal{R}) = \mathcal{R}_{13}\mathcal{R}_{23}$

$$\begin{aligned}(\Delta \otimes \mathrm{id})(\mathcal{R}) &= (\Delta \otimes \mathrm{id})(1 \otimes 1 - \psi_+ \otimes \psi_-) \\ &= 1 \otimes 1 \otimes 1 - (\psi_+ \otimes 1 + 1 \otimes \psi_+) \otimes \psi_- \\ &= 1 \otimes 1 \otimes 1 - \psi_+ \otimes 1 \otimes \psi_- - 1 \otimes \psi_+ \otimes \psi_-.\end{aligned}$$

$$\begin{aligned}\mathcal{R}_{13}\mathcal{R}_{23} &= (1 \otimes 1 \otimes 1 - \psi_+ \otimes 1 \otimes \psi_-)(1 \otimes 1 \otimes 1 - 1 \otimes \psi_+ \otimes \psi_-) \\ &= 1 \otimes 1 \otimes 1 - \psi_+ \otimes 1 \otimes \psi_- - 1 \otimes \psi_+ \otimes \psi_-.\end{aligned}$$

6. $(\mathrm{id} \otimes \Delta)(\mathcal{R}) = \mathcal{R}_{13}\mathcal{R}_{12}$

$$\begin{aligned}(\mathrm{id} \otimes \Delta)(\mathcal{R}) &= (\mathrm{id} \otimes \Delta)(1 \otimes 1 - \psi_+ \otimes \psi_-) \\ &= 1 \otimes 1 \otimes 1 - \psi_+ \otimes (\psi_- \otimes 1 + 1 \otimes \psi_-) \\ &= 1 \otimes 1 \otimes 1 - \psi_+ \otimes \psi_- \otimes 1 - \psi_+ \otimes 1 \otimes \psi_-.\end{aligned}$$

$$\begin{aligned}\mathcal{R}_{13}\mathcal{R}_{12} &= (1 \otimes 1 \otimes 1 - \psi_+ \otimes 1 \otimes \psi_-)(1 \otimes 1 \otimes 1 - \psi_+ \otimes \psi_- \otimes 1) \\ &= 1 \otimes 1 \otimes 1 - \psi_+ \otimes \psi_- \otimes 1 - \psi_+ \otimes 1 \otimes \psi_-.\end{aligned}$$

Thus $\mathcal{H}$ is a quasi-triangular Hopf superalgebra. We define $u := (\nabla \circ \tau \circ (\mathrm{id} \otimes S))(\mathcal{R}) = 1 - \psi_-\psi_+$. We find that $S(u) = S(1 - \psi_-\psi_+) = 1 + \psi_+\psi_- = 1 - \psi_-\psi_+ = u$, and thus we may fix the ribbon element v, a square root of $S(u)u$, as u itself. We check that setting $v = u$ gives a ribbon Hopf superalgebra:

1. v is central in $\mathcal{H}$

$$\begin{aligned}v\psi_\pm &= (1 - \psi_-\psi_+)\psi_\pm \\ &= \psi_\pm \\ &= \psi_\pm(1 - \psi_-\psi_+) \\ &= \psi_\pm v.\end{aligned}$$

2. $v^2 = S(u)u$ (follows from the definition of v).

3. $\Delta(v)(\tau(\mathcal{R})\mathcal{R}) = v \otimes v$

$$\begin{aligned}\Delta(v) &= \Delta(1 - \psi_-\psi_+) \\ &= 1 \otimes 1 - (\psi_- \otimes 1 + 1 \otimes \psi_-)(\psi_+ \otimes 1 + 1 \otimes \psi_+) \\ &= 1 \otimes 1 - \psi_-\psi_+ \otimes 1 - \psi_- \otimes \psi_+ + \psi_+ \otimes \psi_- - 1 \otimes \psi_-\psi_+.\end{aligned}$$

$$\begin{aligned}\tau(\mathcal{R})\mathcal{R} &= (1 \otimes 1 + \psi_- \otimes \psi_+)(1 \otimes 1 - \psi_+ \otimes \psi_-) \\ &= 1 \otimes 1 - \psi_+ \otimes \psi_- + \psi_- \otimes \psi_+ + \psi_-\psi_+ \otimes \psi_+\psi_-.\end{aligned}$$

$$\begin{aligned}\Delta(u)(\tau(\mathcal{R})\mathcal{R}) &= 1 \otimes 1 - \psi_-\psi_+ \otimes 1 - \psi_- \otimes \psi_+ + \psi_+ \otimes \psi_- - 1 \otimes \psi_-\psi_+ - \psi_+ \otimes \psi_- \\ &\quad - \psi_-\psi_+ \otimes \psi_+\psi_- + \psi_- \otimes \psi_+ - \psi_+\psi_- \otimes \psi_-\psi_+ + \psi_-\psi_+ \otimes \psi_+\psi_- \\ &= 1 \otimes 1 - \psi_-\psi_+ \otimes 1 - 1 \otimes \psi_-\psi_+ - \psi_+\psi_- \otimes \psi_-\psi_+.\end{aligned}$$

6.3 Representations of $\mathfrak{psl}(1|1)$

We consider the following three irreducible representations of $\mathfrak{psl}(1|1)$:

1. $W_+ = \mathbb{C}[\psi_+] = \mathbb{C}\langle 1, \psi_+ \rangle$

$$\psi_- = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \psi_+ = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

2. $W_- = \mathbb{C}[\psi_-] = \mathbb{C}\langle 1, \psi_- \rangle$

$$\psi_- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}, \quad \psi_+ = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

3. $P = \mathcal{H} = \mathbb{C}\langle 1, \psi_+, \psi_-, \psi_+\psi_- \rangle$

$$\psi_- = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, \quad \psi_+ = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Definition 6.3.1. Let $\rho : \mathcal{H} \to \mathrm{End}(V)$ be a representation of a Hopf (super)algebra $\mathcal{H}$. The *dual representation* $\rho_{V^*} : \mathcal{H} \to \mathrm{End}(V^*)$ is given by

$$(\rho_{V^*}(x))(f) = f \circ \rho_V(S(a)), \quad \text{for any } x \in \mathcal{H},\ f \in V^*.$$

Let $\{e_i\}$ and $\{e_i^*\}$ be bases for V and V^* respectively. If we represent a vector $v \in V$ by a column vector

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix},$$

we may represent an element $f \in V^*$ as a row vector

$$f = \begin{pmatrix} f_1 & \cdots & f_n \end{pmatrix},$$

so that $f(v)$ is give by $f \circ v = \sum_i f_i v_i \in \mathbb{C}$. Then, given a linear map $A \in \mathrm{End}(V)$, we have

$$f \circ A = \begin{pmatrix} f_1 & \cdots & f_n \end{pmatrix} A = A^T \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix} \text{ (if instead we represent } f \text{ as a column vector).}$$

Therefore, the dual representations $\rho_{V^*}(\psi_\pm)$ will be given by $(\rho_V(S(\psi_\pm))^T$. For the three representations given above, we calculate their dual representations as follows

1. W_+^*

$$\psi_-^* = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad \psi_+^* = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}.$$

2. W_-^*

$$\psi_-^* = \begin{pmatrix} 0 & -1 \\ 0 & 0 \end{pmatrix}, \quad \psi_+^* = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}.$$

3. P^*

$$\psi_-^* = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad \psi_+^* = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

As in section 5, for each representation ρ of $\mathcal{H}$ on a ($\mathbb{Z}_2$-graded) vector space V, we define $R \in \mathrm{End}(V \otimes V)$ and $h \in \mathrm{End}(V)$ by

$$R = \tau \circ ((\rho_V \otimes \rho_V)(\mathcal{R})), \quad h = \rho_V(uv^{-1}).$$

Recall that $\tau \in \mathrm{End}(V \otimes V)$ is the graded permutation given by

$$\tau(x \otimes y) = (-1)^{|x||y|} y \otimes x, \quad \text{for } x, y \in V,$$

and further, for a graded vector space $V = V_0 \oplus V_1$ we have a natural grading on $V \otimes V$ given by

$$(V \otimes V)_i = \bigoplus_{(j,k):j+k=i} V_j \otimes V_k,$$

where the addition is modulo 2. We consider the representations $W_\pm$ and P of $\mathcal{H}$ in turn.

6.3.1 The representations $W_\pm$

We first note that for the vector spaces $W_\pm = \mathbb{C}\langle 1, \psi_\pm \rangle$, we have that $W_\pm \otimes W_\pm = \mathbb{C}\langle 1 \otimes 1, 1 \otimes \psi_\pm, \psi_\pm \otimes 1, \psi_\pm \otimes \psi_\pm \rangle$. In this basis, the $\mathbb{Z}_2$-graded permutation τ is presented by

$$\tau = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

Since $h = \rho(uv^{-1})$ and $u = v$, h will always be represented by the identity matrix. Further, we calculate R as

$$\tau \circ (\rho_{W_\pm} \otimes \rho_{W_\pm})(\mathcal{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

Unfortunately, we immediately see that $R^{-1} = R$. Thus, any invariant we defined from this wouldn't differentiate between over- and under-crossings. Since every knot has an unknotting number this will give the same value on every knot (and every link with the same number of components).

For a 2-component link, we may associate the representation W_+ to one component, and W_- to the other component (the 'mixing' of representations here only makes sense for links, where each strand is represented by a copy of a (possibly different) vector space). We may then calculate

$$\tau \circ (\rho_{W_-} \otimes \rho_{W_+})(\mathcal{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

$$\tau \circ (\rho_{W_+} \otimes \rho_{W_-})(\mathcal{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix}.$$

However, both of these R-matrices are still equal to their own inverses, so any invariant defined using them will be trivial.

6.3.2 The representation P

As in the previous section, the endomorphism h is given by the identity matrix. The endomorphism R is calculated as

$$\tau \circ (\rho_P \otimes \rho_P)(\mathcal{R}) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

Using the maps R and h above, as in section 4.1, we try to obtain an invariant by taking the trace of $h^{\otimes n} \cdot \psi_n(b)$. However, since we are working with $\mathbb{Z}_2$-graded vector space (and permutation map), we will need to define a slightly modified trace.

Definition 6.3.2. Let V be a $\mathbb{Z}_2$-graded vector space with basis $\{e_i\}$, and let $A \in \mathrm{End}(V)$. The *supertrace* of A, denoted by $\mathrm{str}(A)$, is given by

$$\mathrm{str}(A) = \sum_i (-1)^{|e_i|} A_{ii}.$$

However, it turns out that for the representations given above, the invariant $\mathrm{str}(h^{\otimes n} \cdot \psi_n(b))$ of a link with braid representation b will always be zero. Thus, analogously to the calculation section 4.1.2, we take a modified partial trace.

Definition 6.3.3. Let V_1, V_2 be super vector spaces over $\mathbb{C}$ with bases $\{e_i\}$ and $\{e'_i\}$ respectively, and let $A \in \mathrm{End}(V_1 \otimes V_2)$ be presented by $A(e_i \otimes e'_j) = \sum_{k,l} A_{ij}^{kl} e_k \otimes e'_l$. Then the *partial supertraces* $\mathrm{str}_1(A)$ and $\mathrm{str}_2(A)$ are given by

$$(\mathrm{str}_1(A))_{ij} = \sum_k (-1)^{|e_k|} A_{ki}^{kj}$$

$$(\mathrm{str}_2(A))_{ij} = \sum_k (-1)^{|e'_k|} A_{ik}^{jk}$$

It can be shown that the R-matrix given above is not equal to its own inverse. However, it does seem to give the same value (1) on all knots up to 7_2, and 0 on all links up to L6n1.[4]

[4]Knots and links with braid presentations of more than 3 strands were excluded, because calculations become too slow (with 256 dimensional matrices). This excluded the knot 6_1 and the links L4a1, L6a1, L6a2, L6a5. See A.5 for calculations. Braid presentations and naming conventions for knots and links are due to [1].

A Code

In this appendix, we provide the code (written in SageMath) used for testing and implementing various formulas and algorithms given in the main body of the text. Sections A.1 and A.4 define basic functions needed for later calculations, A.2 explores the calculations done in section 4, A.3 verifies calculations in 5 and A.5 explores the invariants defined in section 6.

A.1 Preliminary functions

```
# this is a comment
#this is a test

t = var('t')
R_jones = Matrix(SR, 4, 4, [t^(1/2),0,0,0, 0,0,t,0, 0,t,t^(1/2)-t^(3/2),0,
0,0,0,t^(1/2)])
R_alex = Matrix(SR, 4, 4, [t^(-1/2),0,0,0, 0,0,1,0, 0,1,t^(-1/2)-t^(1/2),0,
0,0,0,-t^(1/2)])
h_jones = Matrix(SR, 2, 2, [t^(-1/2),0, 0,t^(1/2)])
h_alex = Matrix(SR, 2, 2, [t^(1/2),0, 0,-t^(1/2)])

def t_prod(A, B):
    '''
    Parameters
    ----------
    A : Matrix
        first matrix.
    B : Matrix
        second matrix

    Returns
    -------
    AtB_final : Matrix
        kronecker product of A and B
    '''
    m=A.nrows()
    n=A.ncols()
    p=B.nrows()
    q=B.ncols()
    AtB_list = []

    for i in range(m):
        ith_blocks = []

        for j in range(n):
            AijB=A[i,j]*B
            ith_blocks.append(AijB)

        for k in range(p):
            for block in ith_blocks:
                AtB_list.append(block[k])
```

```
    flat = [item for sublist in AtB_list for item in sublist]
    AtB_final = Matrix(SR, p*m, q*n, flat)

    return AtB_final

def nth_tens(A, n):
    '''
    Parameters
    ----------
    A : Matrix
        matrix to tensor.
    n : int
        power of tensor product

    Returns
    -------
    newA : Matrix
        A tensored with itself n times
    '''
    newA = A
    if n == 0:
        return Matrix(SR, 1, 1, [1])
    for i in range(n-1):
        newA = t_prod(newA,A)
    return newA

def prods(l):
    '''
    Parameters
    ----------
    l : list
        list of matrices to tensor.

    Returns
    -------
    Matrix
        the tensor product of the matrices in l
    '''
    while len(l) > 1:
        l_new = l[:-2]
        last = t_prod(l[-2], l[-1])
        l_new.append(last)
        l = l_new

    return l[0]

def nth_permutation(n):
    '''
    Parameters
    ----------
```

```
    n : int
        dimension of V

    Returns
    -------
    P : Matrix
        permutation matrix in End(VxV)
        e_i x e_j --> e_j x e_i
    '''
    P = Matrix(SR, n^2, n^2)
    while P[n^2 - 1, n^2 - 1] != 1:
        for i in range(n):
            P[i*n, i] = 1
            for j in range(n):
                P[i*n + j, i + j*n] = 1

    return P
```

A.2 Braids and tangles

```
def braid(el, n, unity):
    '''
    Parameters
    ----------
    el : list
        list of generators forming the braid rep
        e.g. (s_1) (s_2)^(-1) (s_3)^2 (s_2) --> [1, -2, 3, 3, 2]
    n : int
        number of strands (el in B_n)
    unity : bool
        True if Alexander polynomial, False if Jones polynomial

    Returns
    -------
    poly : expr
        Alexander polynomial if unity
        ((t^(1/2)+t^(-1/2)) * Jones polynomial) otherwise
    '''
    prod = []
    idd = identity_matrix(2)
    for i in el:
        inv = False
        if i < 0:
            inv = True
            i = abs(i)
        l = nth_tens(idd, i-1)
        r = nth_tens(idd, n-i-1)

        if inv:
            if unity:
                phi = t_prod(l, t_prod(R_alex.inverse(), r))
```

```
                else:
                    phi = t_prod(l, t_prod(R_jones.inverse(), r))
            else:
                if unity:
                    phi = t_prod(l, t_prod(R_alex, r))
                else:
                    phi = t_prod(l, t_prod(R_jones, r))
            prod.append(phi)

        phib = identity_matrix(2**n)
        for p in prod:
            phib = phib*p

        if unity:
            hphi = t_prod(idd, nth_tens(h_alex, n-1))*phib
            poly = almost_trace(hphi)
            if not bool(poly[0,0] == poly[1,1]):
                print("whoops!") # checks the matrix is a multiple of id_V
                return None
            else:
                poly = poly[0,0]
        else:
            hphi = nth_tens(h_jones, n)*phib
            poly = hphi.trace()

        return (poly.expand()).simplify()

def almost_trace(A):
    '''
    Parameters
    ----------
    A : Matrix
        2^n x 2^n matrix (in End(VxVx...xV)) to 'almost trace' over
        (see Returns)

    Returns
    -------
    M : matrix
        trace_{2, 3, ..., n}(A)
    '''
    n=A.nrows()
    m = int(n/2)
    M = Matrix(SR, 2, 2)
    for i in range(2):
        for j in range(2):
            M[i,j] = sum([A[k+i*m,k+j*m] for k in range(m)])
    return M

factor = t^(1/2)+t^(-1/2)
```

```
# 3_1
#pretty_print(braid([1, 1, 1], 2, False))
#pretty_print((factor*(t+t^3-t^4)).expand())
#pretty_print(braid([1, 1, 1], 2, True))
#pretty_print((t-1+(t^-1)).expand())

# 4_1
#pretty_print(braid([1, -2, 1, -2], 3, False))
#pretty_print((factor*(t^2-t+1-t^(-1)+t^(-2))).expand())
#pretty_print(braid([1, -2, 1, -2], 3, True))
#pretty_print((-t^(-1)+3-t).expand())

# 5_1
#pretty_print(braid([-1, -1, -1, -1, -1], 2, False))
#pretty_print((factor*(-t^(-7)+t^(-6)-t^(-5)+t^(-4)+t^(-2))).expand())
#pretty_print(braid([-1, -1, -1, -1, -1], 2, True))
#pretty_print((t^2+t^(-2)-t-t^(-1)+1).expand())

# 5_2
#pretty_print(braid([-1, -1, -2, -2, -1, 2], 3, False))
#x = (factor*(-t^(-6)+t^(-5)-t^(-4)+2*(t^(-3))-t^(-2)+t^(-1))).expand()
#pretty_print(x)
#pretty_print(braid([-1, -1, -2, -2, -1, 2], 3, True))
#pretty_print((2*t+2*t^(-1)-3).expand())

# Jones
n = Matrix(SR, 1, 4, [t^(-1/2),0,0,t^(1/2)])
n_ = Matrix(SR, 1, 4, [1,0,0,1])
u = Matrix(SR, 4, 1, [t^(1/2),0,0,t^(-1/2)])
u_ = Matrix(SR, 4, 1, [1,0,0,1])

idd = identity_matrix(2)

# 3_1 jones - cups and caps
a1 = t_prod(n_, n)
a2 = t_prod(idd, t_prod(R_jones, idd))
a3 = t_prod(u, u_)

f = a1*a2*a2*a2*a3
#pretty_print(f.expand())

# 4_1 jones - cups and caps
a1 = t_prod(n_, n)
a2 = prods([idd, R_jones, idd])
a3 = prods([idd, idd, idd, n, idd])
a4 = prods([idd, idd, R_jones.inverse(), idd, idd])
a5 = prods([idd, R_jones, idd, idd, idd])
a6 = prods([idd, idd, idd, u_, idd])
a7 = t_prod(u, u_)

f = a1*a2*a3*a4*a5*a4*a6*a7
```

```
#pretty_print(f.expand())

# 5_1 jones
a1 = t_prod(n_, n)
a2 = prods([idd, R_jones.inverse(), idd])
a3 = t_prod(u, u_)

f = a1*(a2^5)*a3
#pretty_print(f.expand())

# 5_2 jones
a1 = t_prod(n_, n)
a2 = prods([idd, idd, idd, n, idd])
a3 = prods([idd, R_jones.inverse(), idd, idd, idd])
a4 = prods([idd, idd, R_jones.inverse(), idd, idd])
a5 = prods([idd, idd, R_jones, idd, idd])
a6 = prods([idd, idd, idd, u_, idd])
a7 = t_prod(u, u_)

f = a1*a2*(a3^2)*(a4^2)*a3*a5*a6*a7
#pretty_print(f.expand())

# ALexander
n = Matrix(SR, 1, 4, [t^(1/2),0,0,-t^(1/2)])
n_ = Matrix(SR, 1, 4, [1,0,0,1])
u = Matrix(SR, 4, 1, [-t^(1/2),0,0,t^(1/2)])
u_ = Matrix(SR, 4, 1, [1,0,0,1])

# 3_1 alexander - 1-tangle
a1 = t_prod(idd, n)
a2 = t_prod(R_alex.inverse(), idd)
a3 = t_prod(idd, u_)

f = a1*a2*a2*a2*a3
#pretty_print(f[0, 0].expand())

# 4_1 alexander - 1-tangle
a1 = t_prod(idd, n)
a2 = t_prod(R_alex, idd)
a3 = prods([idd, idd, n, idd])
a4 = prods([idd, R_alex.inverse(), idd, idd])
a5 = prods([R_alex, idd, idd, idd])
a6 = prods([idd, idd, u_, idd])
a7 = t_prod(idd, u_)

f = a1*a2*a3*a4*a5*a4*a6*a7
#pretty_print(f[0,0].expand())

# 5_1 alexander - 1-tangle
a1 = t_prod(idd, n)
```

```
a2 = t_prod(R_alex.inverse(), idd)
a3 = t_prod(idd, u_)

f = a1*(a2^5)*a3
#pretty_print(f[0,0].expand())

# 5_2 alexander
a1 = t_prod(idd, n)
a2 = prods([idd, idd, n, idd])
a3 = prods([R_alex.inverse(), idd, idd, idd])
a4 = prods([idd, R_alex.inverse(), idd, idd])
a5 = prods([idd, R_alex, idd, idd])
a6 = prods([idd, idd, u_, idd])
a7 = t_prod(idd, u_)

f = a1*a2*(a3^2)*(a4^2)*a3*a5*a6*a7
#pretty_print(f[0,0].expand())
```

A.3 $U_q(\mathfrak{sl}_2)$ and $U_\zeta(\mathfrak{sl}_2)$

```
q = var('q')
xi = var('x')
lam = var('l')
# sage won't simplify expressions if given the variables as unicodes
# so i redefine these at the end

def quantum(qq, n):
    '''
    Parameters
    ----------
    qq : var
        q or xi (depending on unity).
    n : int
        integer to quantise.

    Returns
    -------
    bracket : expr
        [n]
    '''
    bracket = (qq^(n/2) - qq^(-n/2))/(qq^(1/2)-qq^(-1/2))
    return bracket

def quantum_fact(qq, n):
    '''
    Parameters
    ----------
    qq : var
        q or xi (depending on unity).
    n : int
        integer to quantum factorial.
```

```
    Returns
    -------
    p : expr
        [n]!.
    '''
    p = 1
    for i in range(1, n+1):
        p*= quantum(qq, i)
    return p

def exp_qq(qq, x, n):
    '''
    Parameters
    ----------
    qq : var
        q or xi (depending on unity).
    x : expr
        expression to exponentiate.
    n : int
        upper limit (dimension or r : xi^r = 1).

    Returns
    -------
    s : expr
        exp_qq (x).
    '''
    s = 0
    for i in range(n):
        s+= ((qq^(i*(i-1)/4))/(quantum_fact(qq, i)))*(x^i)
    return s

def p_Vn_E(n, unity):
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n-dimensional rho(E).
    '''
    E = Matrix(SR, n, n)
    for i in range(n):
        if i < n-1:
            if unity:
```

```
                E[i,i+1] = quantum(xi, n-i-1+lam)
            else:
                E[i,i+1] = quantum(q, n - i-1)

    if unity:
        return E
    else:
        return E.simplify_full()

def p_Vn_F(n, unity):
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n-dimensional rho(F).
    '''
    F = Matrix(SR, n, n)
    for i in range(n):
        if i > 0:
            if unity:
                F[i,i-1] = quantum(xi, i)
            else:
                F[i,i-1] = quantum(q, i)
    return F.simplify_full()

def p_Vn_K(n, unity):
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n-dimensional rho(K).
    '''
    K = Matrix(SR, n, n)
    for i in range(n):
        if unity:
            K[i,i] = xi^((n-(2*i+1)+lam)/2)
```

```
        else:
            K[i,i] = q^((n-(2*i+1))/2)

    if unity:
        return K
    else:
        return K.simplify_full()

def p_Vn_H(n, unity):
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n-dimensional rho(H).
    '''
    H = Matrix(SR, n, n)
    for i in range(n):
        if unity:
            H[i,i] = n-(2*i+1)+lam
        else:
            H[i,i] = n-(2*i+1)
    return H

def qq_pow_mat(qq, diag):
    '''
    Parameters
    ----------
    qq : var
        q or xi (depending on unity).
    diag : Matrix
        diagonal matrix to exponentiate.

    Returns
    -------
    diag : Matrix
        qq^diag.
    '''
    for i in range(diag.nrows()):
        diag[i,i] = qq^(diag[i,i])

    return diag

def R(n, unity):
```

```
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n^2-dimensional (rho x rho)(R).
    '''
    # variables below are so that the equation doesn't run over the page
    po = (1/4)*t_prod(p_Vn_H(n, unity), p_Vn_H(n, unity))
    af = t_prod(p_Vn_E(n, unity), p_Vn_F(n, unity))
    if unity:
        R = qq_pow_mat(xi,po)*exp_qq(xi,(xi^(1/2)-xi^(-1/2))*af,n)
        return R
    else:
        R = qq_pow_mat(q,po)*exp_qq(q,(q^(1/2)-q^(-1/2))*af,n)
        return R.simplify_full()

def u(n, unity):
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n-dimensional rho(u).
    '''
    s = 0
    for i in range(n):
        if unity:
            fr = ((xi^(-1/2)-xi^(1/2))^i)/quantum_fact(xi,i)
            af = (p_Vn_K(n, unity)^(-i))*(p_Vn_E(n,unity)^i)
            s += xi^(3*i*(i-1)/4)*fr*(p_Vn_F(n,unity)^i)*af
        else:
            fr = ((q^(-1/2)-q^(1/2))^i)/quantum_fact(q, i)
            af = (p_Vn_K(n, unity)^(-i))*(p_Vn_E(n, unity)^i)
            s += q^(3*i*(i-1)/4)*fr*(p_Vn_F(n, unity)^i)*af

    if unity:
        uu = qq_pow_mat(xi, (-1/4)*(p_Vn_H(n, unity)^2))*s
```

```
        return uu.simplify_full()
    else:
        uu = qq_pow_mat(q, (-1/4)*(p_Vn_H(n, unity)^2))*s
        return uu.simplify_full()

def v(n, unity):
    '''
    Parameters
    ----------
    n : int
        dimension or r (depending on unity).
    unity : bool
        True if xi is a root of unity (alexander), false otherwise (jones).

    Returns
    -------
    Matrix
        n-dimensional rho(v).
    '''
    if unity:
        vv = (p_Vn_K(n, unity)^(n-1))*u(n, unity)
    else:
        vv = (p_Vn_K(n, unity)^(-1))*u(n, unity)
    return vv

P = Matrix(SR, 4, 4, [1,0,0,0, 0,0,1,0, 0,1,0,0, 0,0,0,1])
R_jones = P*R(2, False)
h_jones = p_Vn_K(2, False)
#pretty_print(R_jones)
#pretty_print(h_jones)

l = var('\u03BB')
x = var('\u03B6') # right letter now...

R_alex = (P*R(2, True)).subs(xi==x, lam==l)
h_alex = (p_Vn_K(2, True)^(1-2)).subs(xi==x, lam==l) #1-r
#pretty_print(R_alex)
#pretty_print(h_alex)

normal_jones = v(2, False)[0,0]
normal_alex = (v(2, True)[0,0]).simplify_full().subs(xi==x, lam==l)
#pretty_print(normal_jones)
#pretty_print(normal_alex)
```

A.3.1 Skein relations

```
def skein_checker(n):
    '''
    Parameters
    ----------
    n : int
```

```
        dimension of V

    Returns
    -------
    dimension : int
        dimension of the subspace of elements which commute with the
        action of U_q(sl_2)
    '''
    I = identity_matrix(n)
    E = p_Vn_E(n, False)
    F = p_Vn_F(n, False)
    K = p_Vn_K(n, False)

    s = ""
    for i in range(1, n^4 + 1):
        s += " a" + str(i)
    s = s[1:]
    vv = var(s)
    general_mat = Matrix(SR, n^2, n^2, list(vv))
    eqns = []
    delts = [t_prod(I,I),t_prod(E,K)+t_prod(I,E)]
    delts.append(t_prod(F,I)+t_prod(K^(-1),F))
    delts.append(t_prod(K,K))
    for d in delts:
        dF = d*general_mat
        Fd = general_mat*d
        for i in range(n^2):
            for j in range(n^2):
                eqns.append(dF[i,j]==Fd[i,j])

    sols = solve(eqns,vv)[0]
    vs = [i.variables() for i in sols]
    flat_vs = [i for j in vs for i in j]
    ar_set = set(flat_vs)
    r_set = set([el for el in ar_set if str(el)[0] == "r"])

    #return(solve(eqns, vv)) # uncomment to see the form of the solutions
    dimension = len(r_set)
    return dimension

#pretty_print(skein_checker(2))
#pretty_print(skein_checker(3))
```

A.4 Preliminary superfunctions

```
def supertrace(M, parity_basis_V):
    '''
    Parameters
    ----------
    M : matrix
        matrix to supertrace over
```

```
    parity_basis_V : list
        the parity of each element of the basis of V
        (e.g. [0,1] for V = C<1,psi>)

    Returns
    -------
    tr : expr
        the supertrace of M
    '''
    m = len(parity_basis_V)
    n = M.nrows()
    tr = 0
    for i in range(n):
        if m^i == n:
            break
    parity_basis = parity_basis_Vn(parity_basis_V, i)
    for j in range(n):
        tr += ((-1)^(parity_basis[j]))*M[j,j]
    return tr

def almost_supertrace(M, parity_basis_V):
    '''
    Parameters
    ----------
    M : matrix
        matrix to "almost supertrace" over
    parity_basis_V : list
        the parity of each element of the basis of V
        (e.g. [0,1] for V = C<1,psi>)

    Returns
    -------
    Str2 : expr
        str_{2,3,...,n}(M)
    '''
    m = len(parity_basis_V)
    n = 0
    while m^n < M.nrows():
        n+=1
    parity_basis2 = parity_basis_Vn(parity_basis_V, n-1)
    Str2 = Matrix(SR, m, m)
    for i in range(m):
        for j in range(m):
            s = 0
            for k in range(m^(n-1)):
                Mi = k + i*int(M.nrows()/m)
                Mj = k + j*int(M.nrows()/m)
                parity = parity_basis2[k]
                s += ((-1)^parity)*M[Mi,Mj]
            Str2[i,j] = s
```

```
    return Str2

def parity_basis_Vn(parity_basis_V, n):
    '''
    Parameters
    ----------
    parity_basis_V : list
        the parity of each element of the basis of V
        (e.g. [0,1] for V = C<1,psi>)
    n : int
        number of times V has been tensored with itself

    Returns
    -------
    pb : list
        a list of the parity of each element of the basis of V^{(x)n}
    '''
    m = len(parity_basis_V)
    parity_basis = parity_basis_V
    while len(parity_basis) < m^n:
        new_parity_basis = []
        for el1 in parity_basis:
            for el2 in parity_basis_V:
                new_parity_basis.append(el1+el2)
        parity_basis = new_parity_basis

    pb = [i % 2 for i in parity_basis]

    return pb

def super_nth_permutation(parity_basis_V):
    '''
    Parameters
    ----------
    parity_basis_V : list
        the parity of each element of the basis of V
        (e.g. [0,1] for V = C<1,psi>)

    Returns
    -------
    P : Matrix
        super permutation matrix in End(VxV)
        e_i x e_j --> (-1)^(|e_j||e_i|) e_j x e_i
    '''
    n = len(parity_basis_V)
    P = Matrix(SR, n^2, n^2)
    pb = []
    for i in parity_basis_V:
        for j in parity_basis_V:
```

```
            pb.append(i+j)
    npb = []
    for p in pb:
        if p == 2:
            npb.append(1)
        else:
            npb.append(0)
    while P[n^2 - 1, n^2 - 1] == 0:
        for i in range(n):
            P[i*n, i] = (-1)^(npb[i*n]*npb[i])
            for j in range(n):
                P[i*n + j, i + j*n] = (-1)^(npb[i*n + j]*npb[i + j*n])

    return P
```

A.5 $\mathfrak{psl}(1|1)$

```
def psi_pos(pos, P):
    '''
    Parameters
    ----------
    pos : bool
        True if W+ rep, False is W- rep (doesn't matter if P rep)
    P : bool
        True if P rep, False otherwise

    Returns
    -------
    psi : Matrix
        rep of psi+
    '''
    if P:
        psi = Matrix(SR, 4, 4, [0,0,0,0, 1,0,0,0, 0,0,0,0, 0,0,1,0])
        return psi
    else:
        if pos:
            psi = Matrix(SR, 2, 2, [0,0, 1,0])
        else:
            psi = zero_matrix(2)

        return psi

def psi_neg(pos, P):
    '''
    Parameters
    ----------
    pos : bool
        True if W+ rep, False is W- rep (doesn't matter if P rep)
    P : bool
        True if P rep, False otherwise
```

```
    Returns
    -------
    psi : Matrix
        rep of psi-
    '''
    if P:
        psi = Matrix(SR, 4, 4, [0,0,0,0, 0,0,0,0, 1,0,0,0, 0,-1,0,0])
        return psi
    else:
        return psi_pos(not(pos), False)

def D(Mat):
    '''
    Parameters
    ----------
    Mat : matrix
        matrix to find the coproduct of

    Returns
    -------
    M : Matrix
        Delta(Mat)
    '''
    I = identity_matrix(Mat.nrows())
    M = t_prod(Mat, I) + t_prod(I, Mat)
    return M

def S(Mat):
    '''
    Parameters
    ----------
    Mat : matrix
        matrix to find the antipode of

    Returns
    -------
    Matrix
        S(Mat)
    '''
    return (-1)*Mat

def Rm(pos, P):
    '''
    Parameters
    ----------
    pos : bool
        True if W+ rep, False is W- rep (doesn't matter if P rep)
    P : bool
        True if P rep, False otherwise
```

```
    Returns
    -------
    Rmat : Matrix
        rep of R
    '''
    psi_p = psi_pos(pos, P)
    psi_n = psi_neg(pos, P)
    d = (psi_p.nrows())^2
    if P:
        parity_basis_V = [0,1,1,0] # V = <1,p,n,pn>
    else:
        parity_basis_V = [0,1] # V = <1,psi>
    sP = super_nth_permutation(parity_basis_V)
    Rmat = sP*(identity_matrix(d) - t_prod(psi_p, psi_n))
    return Rmat

#pretty_print(Rm(True, False)) # W+ (x) W+
#pretty_print(Rm(False, False)) # W- (x) W-
iddd = identity_matrix(4)
ip = iddd - t_prod(psi_pos(True, False), psi_neg(False, False))
RR = super_nth_permutation([0,1])*ip
#pretty_print(RR) # W+ (x) W-
ip = iddd - t_prod(psi_pos(False, False), psi_neg(True, False))
RR = super_nth_permutation([0,1])*ip
#pretty_print(RR) # W- (x) W+
#pretty_print(Rm(True, True)) # P

def super_skein_checker(pos, P):
    '''
    Parameters
    ----------
    pos : bool
        True if W+ rep, False is W- rep (doesn't matter if P rep)
    P : bool
        True if P rep, False otherwise

    Returns
    -------
    int
        dimension of subspace which commutes w action of psl(1|1)
    '''
    psi_p = psi_pos(pos, P)
    psi_n = psi_neg(pos, P)
    n = psi_n.nrows()
    I = identity_matrix(n)
    els = [psi_p, psi_n, I]

    s = ""
    for i in range(1, n^4 + 1):
        s += " a" + str(i)
```

```
    s = s[1:]
    vv = var(s)
    general_mat = Matrix(SR, n^2, n^2, list(vv))
    eqns = []
    delts = [D(el) for el in els]
    for d in delts:
        dF = d*general_mat
        Fd = general_mat*d
        for i in range(n^2):
            for j in range(n^2):
                eqns.append(dF[i,j]==Fd[i,j])

    sols = solve(eqns,vv)[0]
    vs = [i.variables() for i in sols]
    flat_vs = [i for j in vs for i in j]
    ar_set = set(flat_vs)
    r_set = set([el for el in ar_set if str(el)[0] == "r"])

    #return(solve(eqns, vv))
    return len(r_set)

# below: dimensions of subspaces | D(a).F = F.D(a)
#print(super_skein_checker(True, False)) # W+
#print(super_skein_checker(False, False)) # W-
#print(super_skein_checker(True, True)) # P

def superbraid(el, n, Rmat, hmat, unity, parity_basis_V):
    '''
    Parameters
    ----------
    el : list
        list of generators forming the braid rep
        e.g. (s_1) (s_2)^(-1) (s_3)^2 (s_2) --> [1, -2, 3, 3, 2]
    n : int
        number of strands (el in B_n)
    Rmat : Matrix
        R matrix
    hmat : Matrix
        h matrix
    unity : bool
        True to supertrace, False to just trace
        (supertrace should always be zero, so set this as True!)
        (False option was just for testing)
    party_basis_V : list
        the parity of each element of the basis of V
        (e.g. [0,1] for V = C<1,psi>)

    Returns
    -------
    poly : expr
```

```
        invariant from R and h
    '''
    prod = []
    d = hmat.nrows()
    idd = identity_matrix(d)
    c = 0
    for i in el:
        c += 1
        inv = False
        if i < 0:
            inv = True
            i = abs(i)
        l = nth_tens(idd, i-1)
        r = nth_tens(idd, n-i-1)

        if inv:
             phi = t_prod(l, t_prod(Rmat.inverse(), r))

        else:
            phi = t_prod(l, t_prod(Rmat, r))

        prod.append(phi)

    phib = identity_matrix(d**n)
    for p in prod:
        phib = phib*p

    if unity:
        hphi = t_prod(idd, nth_tens(hmat, n-1))*phib
        poly = almost_supertrace(hphi,parity_basis_V)
        if not bool(poly[0,0] == poly[1,1]):
            print("whoops!") # checks the matrix is a multiple of id_V
            return None
        else:
            poly = poly[0,0]
    else:
        hphi = nth_tens(hmat, n)*phib
        poly = supertrace(hphi, parity_basis_V)

    return (poly.expand()).simplify()

iddd = identity_matrix(4)
b = [1, 1, 1] # 3_1
pretty_print(superbraid(b, 2, Rm(True, True), iddd, True, [0,1,1,0]))
b = [1, -2, 1, -2] # 4_1
pretty_print(superbraid(b, 3, Rm(True, True), iddd, True, [0,1,1,0]))
b = [-1, -1, -1, -1, -1] # 5_1
pretty_print(superbraid(b, 2, Rm(True, True), iddd, True, [0,1,1,0]))
b = [-1, -1, -2, -2, -1, 2] # 5_2
pretty_print(superbraid(b, 3, Rm(True, True), iddd, True, [0,1,1,0]))
```

```
b = [-3, -3, -2, 1, 3, -2, 1] # 6_1
#pretty_print(superbraid(b, 4, Rm(True, True), iddd, True))
b = [-2, -2, -2, 1, -2, 1] # 6_2
pretty_print(superbraid(b, 3, Rm(True, True), iddd, True, [0,1,1,0]))
b = [-2, -2, 1, -2, 1, 1] # 6_3
pretty_print(superbraid(b, 3, Rm(True, True), iddd, True, [0,1,1,0]))
b = [-1, -1, -1, -1, -1, -1, -1] # 7_1
pretty_print(superbraid(b, 2, Rm(True, True), iddd, True, [0,1,1,0]))
b = [-3, -3, -3, -2, 3, -2, -1, 2, -1] #7_2
#pretty_print(superbraid(b, 4, Rm(True, True), iddd, False)) #7_2
# up to 7_2 - braids w 4 strands seem to be too much for my computer
# which is why they're commented out here.
# (Sage is having to deal with 256x256 matrices, to be fair)
```

References

[1] Dror Bar-Natan, Scott Morrison and et al. *The Knot Atlas*. URL: http://katlas.org.

[2] David M. Jackson and Iain Moffatt. *An Introduction to Quantum and Vassiliev Knot Invariants*. 2019.

[3] W.B. Raymond Lickorish. *An Introduction to Knot Theory*. 1997.

[4] Kunio Murasugi. *Knot Theory and Its Applications*. 1996.

[5] Tomotada Ohtsuki. *Quantum Invariants*. 2002.

[6] D.B. Westra. *Structure of Super Hopf Algebras*. www.mat.univie.ac.at. URL: https://www.mat.univie.ac.at/~westra/superhopfalg.pdf (visited on 14/08/2023).